



PMR-3401: Mecânica Computacional para Mecatrônica

Exercício-Programa 2

feito por

João Rodrigo Windisch Olenscki 10773224
Lui Damianci Ferreira 10770579

Professores: Prof. Dr. Emilio Carlos Nelli Silva
Prof. Dr. Flávio Buiochi

27 de junho de 2021

Lista de Figuras

1	Grade proposta	8
2	Distribuição de A_z para $\Delta x = 5 \cdot 10^{-3}$ m	14
3	Perfil de A_z para $\Delta x = 5 \cdot 10^{-3}$ m	14
4	Campo vetorial \mathbf{B} para $\Delta x = 5 \cdot 10^{-3}$ m	15
5	Campo vetorial \mathbf{H} para $\Delta x = 5 \cdot 10^{-3}$ m	15
6	Distribuição de A_z para $\Delta x = 2.5 \cdot 10^{-3}$ m	16
7	Perfil de A_z para $\Delta x = 2.5 \cdot 10^{-3}$ m	17
8	Campo vetorial \mathbf{B} para $\Delta x = 2.5 \cdot 10^{-3}$ m	17
9	Campo vetorial \mathbf{H} para $\Delta x = 2.5 \cdot 10^{-3}$ m	18
10	Distribuição de A_z para $\Delta x = 1 \cdot 10^{-3}$ m	19
11	Perfil de A_z para $\Delta x = 1 \cdot 10^{-3}$ m	19
12	Campo vetorial \mathbf{B} para $\Delta x = 1 \cdot 10^{-3}$ m	20
13	Campo vetorial \mathbf{H} para $\Delta x = 1 \cdot 10^{-3}$ m	20
14	Distribuição de A_z para material não isotrópico e sem regime transitente	22
15	Perfil de A_z para material não isotrópico e sem regime transitente	22
16	Campo vetorial \mathbf{B} para $\Delta x = 1 \cdot 10^{-3}$ m	23
17	Campo vetorial \mathbf{H} para $\Delta x = 1 \cdot 10^{-3}$ m	23
18	Distribuição de A_z para material não isotrópico e em regime transitente	25
19	Campo vetorial \mathbf{B} para os instantes de interesse	26
20	Campo vetorial \mathbf{H} para os instantes de interesse	27
21	Progressão da força $F_{ela,x}$ no tempo	27

Sumário

1	Introdução	5
2	Equacionamentos	5
2.1	Analítico	5
2.1.1	Fronteira externa	5
2.1.2	Interface entre dois meios	6
2.2	Desenvolvimento Numérico	7
2.2.1	Regime Permanente	7
2.2.2	Regime Transiente	10
2.3	Cálculo dos campos B e H	12
2.4	Cálculo da força eletromagnética na armadura	12
3	Itens a), b), c), e d)	13
3.1	$\Delta x = 5 \cdot 10^{-3}$ m	13
3.1.1	Análise do potencial eletromagnético A_z	13
3.1.2	Análise do campo vetorial B	14
3.1.3	Análise do campo vetorial H	15
3.1.4	Análise da força eletromagnética F_{ela}	16
3.2	$\Delta x = 2.5 \cdot 10^{-3}$ m	16
3.2.1	Análise do potencial eletromagnético A_z	16
3.2.2	Análise do campo vetorial B	17
3.2.3	Análise do campo vetorial H	18
3.2.4	Análise da força eletromagnética F_{ela}	18
3.3	$\Delta x = 1 \cdot 10^{-3}$ m	18
3.3.1	Análise do potencial eletromagnético A_z	18
3.3.2	Análise do campo vetorial B	20
3.3.3	Análise do campo vetorial H	20
3.3.4	Análise da força eletromagnética F_{ela}	21
3.4	Análise da influência da discretização	21
4	Item e.1)	21
4.1	Análise do potencial eletromagnético A_z	22
4.2	Análise do campo vetorial B	23
4.3	Análise do campo vetorial H	23
4.4	Análise da força eletromagnética F_{ela}	24
5	Item e.2)	24
5.1	Escolha de Δt	24
5.2	Análise do potencial eletromagnético A_z	25
5.3	Análise do campo vetorial B	25

5.4 Análise do campo vetorial \mathbf{H}	26
5.5 Análise da força eletromagnética $F_{ela,x}$	27
6 Conclusões	28
Apêndices	29
A Listing do script em Python	29

1 Introdução

Com o intuito de se estudar o método de diferenças finitas, em particular de para a solução de um sistema de levitação magnético, e entender a distribuição de fluxos magnéticos dentro de um trilho de trem magnético, gerou-se a motivação do presente exercício programa.

Além disso, é interessante se estudar estes métodos computacionais para compará-los com casos reais, com estudos anteriores e com resultados consolidados e conhecidos no meio científico. Além disso, considerar a coerência dos valores encontrados com casos reais, e para isto foi tomado como exemplo o [maglev alemão](#).

Os principais intuios do presente exercício programa são:

- Análise da distribuição do potencial magnético (A_z);
- Análise do campo do vetor densidade superficial de fluxo magnético (\mathbf{B});
- Análise do campo do vetor intensidade de campo magnético (\mathbf{H}); e
- Análise das forças na armadura

Além disto, deve-se realizar simulações para 3 valores distintos de discretização da malha e compará-los. Finalmente, deve-se realizar uma simulação para um material não isotrópico (no qual $\mu_x \neq \mu_y$), tanto no caso de uma corrente nas bobinas que não varia com o tempo quanto para o caso em que há variação temporária da magnitude e sentido da corrente na bobina.

2 Equacionamentos

2.1 Analítico

Definindo \mathbf{A} tal que \mathbf{B} é o rotacional de \mathbf{A} ($\mathbf{B} = \nabla \times \mathbf{A}$), pode-se aplicar a Lei de Ampère e combiná-la com a definição de \mathbf{H} , vetor de intensidade do campo magnético, em função de \mathbf{B} :

$$\mathbf{H} = \frac{1}{\mu} \mathbf{B} \quad (1)$$

$$\nabla \times \mathbf{H} = \mathbf{J} \Rightarrow \nabla \times \frac{1}{\mu} \mathbf{B} = \mathbf{J} \quad (2)$$

Como se trata de um problema em duas dimensões (x e y), podemos reescrever esta equação da seguinte maneira, em forma de uma equação diferencial parcial (EDP):

$$\frac{\partial^2 A_z}{\partial x^2} + \frac{\partial^2 A_z}{\partial y^2} = -\mu J_z \quad (3)$$

2.1.1 Fronteira externa

A equação (3) é uma típica equação diferencial elíptica. As condições de contorno dadas para a fronteira externa são as seguintes:

CC.1 $A_z = 0$ no ponto A (ponto de coordenadas (20, 10));

CC.2 $\mathbf{B} \cdot \mathbf{n} = 0$ na fronteira externa, com \mathbf{n} sendo o versor normal a superfície; e

CC.3 $\mathbf{n} \times \mathbf{H} = \mathbf{0}$ na fronteira externa.

A partir da condição **CC.2**, têm-se:

$$\mathbf{B} \cdot \mathbf{n} = 0 \Rightarrow (\nabla \times \mathbf{A}) \cdot \mathbf{n} = 0$$

Calculando este rotacional, levando em conta que $\mathbf{A} = A_z \mathbf{e}_z$:

$$\left(\frac{\partial A_z}{\partial y} \mathbf{e}_x - \frac{\partial A_z}{\partial x} \mathbf{e}_y \right) \cdot \mathbf{n} = 0$$

Devido a geometria do problema, este vetor normal a superfície aponta **ou** na direção x (\mathbf{e}_x) **ou** na direção y (\mathbf{e}_y), de forma que se têm as seguintes equações:

$$\begin{cases} \frac{\partial A_z}{\partial y} = 0 & \text{se } \mathbf{n} = \pm \mathbf{e}_x \\ \frac{\partial A_z}{\partial x} = 0 & \text{se } \mathbf{n} = \pm \mathbf{e}_y \end{cases} \quad (4)$$

Usando, então a condição **CC.3**, têm-se:

$$\mathbf{n} \times \mathbf{H} = \mathbf{0} \Rightarrow \mathbf{n} \times \frac{1}{\mu} \mathbf{B} = \mathbf{0} \Rightarrow \mathbf{n} \times \mathbf{B} = \mathbf{0}$$

Novamente, usando a definição do \mathbf{A} :

$$\mathbf{n} \times (\nabla \times \mathbf{A}) = \mathbf{0}$$

Calculando o rotacional de \mathbf{A} , têm-se então:

$$\mathbf{n} \times \left(\frac{\partial A_z}{\partial y} \mathbf{e}_x - \frac{\partial A_z}{\partial x} \mathbf{e}_y \right) = \mathbf{0}$$

Levando em conta, mais uma vez, as restrições geométricas do problema (que implicam que o vetor normal \mathbf{n} aponta apenas nas direções x e y), têm-se:

$$\begin{cases} \frac{\partial A_z}{\partial x} = 0 & \text{se } \mathbf{n} = \pm \mathbf{e}_x \\ \frac{\partial A_z}{\partial y} = 0 & \text{se } \mathbf{n} = \pm \mathbf{e}_y \end{cases} \quad (5)$$

Juntando os resultados de (4) e (5), têm-se que:

$$\begin{aligned} \text{se } \mathbf{n} = \pm \mathbf{e}_x &\Rightarrow \frac{\partial A_z}{\partial x} = \frac{\partial A_z}{\partial y} = 0 \\ \text{se } \mathbf{n} = \pm \mathbf{e}_y &\Rightarrow \frac{\partial A_z}{\partial x} = \frac{\partial A_z}{\partial y} = 0 \end{aligned} \quad (6)$$

É trivial, então, concluir que, em qualquer lugar da fronteira externa, as derivadas em x e em y são nulas. Desta forma, o valor de A_z deve ser constante em toda esta fronteira. Ora, tem-se que $A_z = 0$ para o ponto A (**CC.1**), pertencente a fronteira. Assim, conclui-se que $A_z = 0$ para qualquer ponto pertencente a fronteira externa.

2.1.2 Interface entre dois meios

Para a interface entre dois meios, têm-se as seguintes condições de contorno:

$$\mathbf{CC.4} \quad \mathbf{B}_1 \cdot \mathbf{n} = \mathbf{B}_2 \cdot \mathbf{n}; \text{ e}$$

$$\mathbf{CC.5} \quad \mathbf{n} \times (\mathbf{H}_1 - \mathbf{H}_2) = \mathbf{0} \text{ com } \mathbf{n} \text{ sendo o versor normal a superfície.}$$

Escrevendo a **CC.4** em termos do vetor **A**, obtém-se:

$$\mathbf{n} \cdot \left(\frac{\partial A_z}{\partial y} \Big|_1, -\frac{\partial A_z}{\partial x} \Big|_1, 0 \right) = \mathbf{n} \cdot \left(\frac{\partial A_z}{\partial y} \Big|_2, -\frac{\partial A_z}{\partial x} \Big|_2, 0 \right)$$

Por se tratar de uma mudança de meio, percebe-se que o versor **n** é o mesmo e, então:

$$\begin{cases} \frac{\partial A_z}{\partial y} & \text{é constante se } \mathbf{n} = \pm \mathbf{e}_x \\ \frac{\partial A_z}{\partial x} & \text{é constante se } \mathbf{n} = \pm \mathbf{e}_y \end{cases} \quad (7)$$

Além disto, escrevendo a **CC.5** em função das componentes do vetor definido **A**, têm-se:

$$\mathbf{n} \times \left(\frac{1}{\mu_1} \frac{\partial A_z}{\partial y} \Big|_1 - \frac{1}{\mu_2} \frac{\partial A_z}{\partial y} \Big|_2, \frac{1}{\mu_2} \frac{\partial A_z}{\partial x} \Big|_2 - \frac{1}{\mu_1} \frac{\partial A_z}{\partial x} \Big|_2, 0 \right) = \mathbf{0}$$

Portanto, pode-se isolar os casos de interfaces horizontais ou verticais, que é o caso que está no problema, e obtém-se:

$$\begin{cases} \frac{\partial A_z}{\partial x} \Big|_1 &= \frac{\mu_1}{\mu_2} \frac{\partial A_z}{\partial x} \Big|_2 \text{ se } \mathbf{n} = \pm \mathbf{e}_x \\ \frac{\partial A_z}{\partial y} \Big|_1 &= \frac{\mu_1}{\mu_2} \frac{\partial A_z}{\partial y} \Big|_2 \text{ se } \mathbf{n} = \pm \mathbf{e}_y \end{cases} \quad (8)$$

2.2 Desenvolvimento Numérico

2.2.1 Regime Permanente

Para a montagem do sistema completo a partir de métodos numéricos para a solução da Equação Diferencial Parcial **(3)**, tomou-se uma generalização de todos os casos: deseja-se o equacionamento para um caso e para isso assume-se que o ponto do qual se está calculando as propriedades é, na verdade, uma fronteira entre 4 meios distintos, sem perda de generalidade. Desta forma, se for um ponto dentro de um meio a equação generalizada se simplifica para o caso de um ponto entre um meio só, se for uma fronteira horizontal é um caso onde os meios de acima e de abaixo são iguais e os meios da direita e da esquerda são distintos, e, analogamente, para uma fronteira vertical é um caso onde os meios da direita e da esquerda são iguais entre si e os de cima e de baixo são diferentes.

Além disso, as grandes relevâncias desta formulação é a simplificação da legibilidade do código e a eficiência ganha nas computações. Isso ocorre pois o processador do computador lida melhor com operações aritméticas do que com comparações, que ocorreriam para a checagem se está dentro de um meio ou em uma fronteira horizontal ou ainda em uma fronteira vertical. Portanto, além de agregar a simplicidade de leitura se agrupa a modularidade para casos genéricos com mais fronteiras que vierem a existir em novas geometrias do problema, a um custo de uma formulação mais complicada para os autores.

Por simplicidade de notação, usará-se a seguinte notação para os pontos, que pode ser observada na Figura 1:

- C* para o ponto acima, correspondendo a *i, j+1*;
- B* para o ponto abaixo, correspondendo a *i, j-1*;
- E* para o ponto a esquerda, correspondendo a *i-1, j*; e
- D* para o ponto a direita, correspondendo a *i+1, j*;

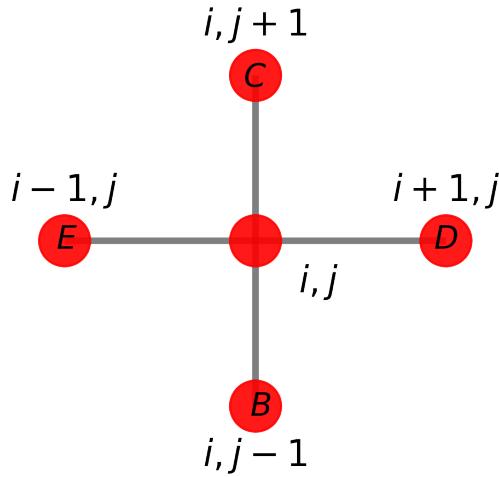


Figura 1: Grade proposta

Os meios apresentam propriedades distintas, que são:

- μ_c para o meio C ;
- μ_b para o meio B ;
- μ_e para o meio E ; e
- μ_d para o meio D

Dessa forma, de maneira análoga ao desenvolvimento realizado para se obter a equação (8), pode-se obter uma relação entre as derivadas parciais em y , que correlaciona os meios c e b , e em x que correlaciona os meios e e d .

$$\begin{cases} \frac{\partial A}{\partial x} \Big|_E &= \frac{\mu_e}{\mu_d} \frac{\partial A}{\partial x} \Big|_D \\ \frac{\partial A}{\partial y} \Big|_C &= \frac{\mu_c}{\mu_b} \frac{\partial A}{\partial y} \Big|_B \end{cases} \quad (9)$$

Então, realizando a expansão em séries de Taylor, até os termos de terceira ordem, para cada um dos sentidos, ou seja, para cada um dos meios, pode-se obter uma expressão para a segunda derivada do vetor \mathbf{A} de acordo com a direção em que se seguiu:

Taylor em C :

$$\mathbf{A}_{i,j+1} = \mathbf{A}_{i,j} + \Delta y \frac{\partial A}{\partial y} \Big|_C + \frac{\Delta y^2}{2} \frac{\partial^2 A}{\partial y^2} \Big|_C$$

$$\frac{\partial^2 A}{\partial y^2} \Big|_C = \frac{2}{\Delta y^2} \left[\mathbf{A}_{i,j+1} - \mathbf{A}_{i,j} - \Delta y \frac{\partial A}{\partial y} \Big|_C \right]$$

Taylor em B :

$$\mathbf{A}_{i,j-1} = \mathbf{A}_{i,j} - \Delta y \frac{\partial A}{\partial y} \Big|_B + \frac{\Delta y^2}{2} \frac{\partial^2 A}{\partial y^2} \Big|_B$$

$$\frac{\partial^2 A}{\partial y^2} \Big|_B = \frac{2}{\Delta y^2} \left[\mathbf{A}_{i,j-1} - \mathbf{A}_{i,j} + \Delta y \frac{\partial A}{\partial y} \Big|_B \right]$$

Taylor em E :

$$\mathbf{A}_{i-1,j} = \mathbf{A}_{i,j} - \Delta x \frac{\partial A}{\partial x} \Big|_E + \frac{\Delta x^2}{2} \frac{\partial^2 A}{\partial x^2} \Big|_E$$

$$\frac{\partial^2 A}{\partial x^2} \Big|_E = \frac{2}{\Delta x^2} \left[\mathbf{A}_{i-1,j} - \mathbf{A}_{i,j} + \Delta x \frac{\partial A}{\partial x} \Big|_E \right]$$

Taylor em D :

$$\mathbf{A}_{i+1,j} = \mathbf{A}_{i,j} + \Delta x \frac{\partial A}{\partial x} \Big|_D + \frac{\Delta x^2}{2} \frac{\partial^2 A}{\partial x^2} \Big|_D$$

$$\frac{\partial^2 A}{\partial x^2} \Big|_D = \frac{2}{\Delta x^2} \left[\mathbf{A}_{i+1,j} - \mathbf{A}_{i,j} - \Delta x \frac{\partial A}{\partial x} \Big|_D \right]$$

É válido ressaltar que para as expansões em série de Taylor realizadas ainda pode-se simplificar ao se montar uma grade quadrada, que é uma das condições impostas pelo enunciado do presente Exercício Programa, que acarreta em $\Delta x = \Delta y$.

Partindo destas equações, pode-se escrever a EDP (3), que relaciona $\frac{\partial^2 A}{\partial x^2}$ e $\frac{\partial^2 A}{\partial y^2}$, que, para o caso mais genérico, será igual a um termo $f|_{i,j}$, que vale 0 nos pontos fora da bobina e não nulo neste meio. Será escrita uma EDP substituindo os termos isolados nas séries de Taylor para cada combinação de elementos na vertical e na horizontal:

- BE ;
- BD ;
- CE ; e
- CD

EDP relacionando B e E

$$\begin{aligned} \frac{2}{\Delta x^2} \left[\mathbf{A}_{i,j-1} + \mathbf{A}_{i-1,j} - 2\mathbf{A}_{i,j} + \Delta x \frac{\partial A}{\partial y} \Big|_B + \Delta x \frac{\partial A}{\partial x} \Big|_E \right] &= f|_{i,j} \\ \frac{\mathbf{A}_{i,j-1} + \mathbf{A}_{i-1,j} - 2\mathbf{A}_{i,j}}{\Delta x} &= -\frac{\partial A}{\partial y} \Big|_B - \frac{\partial A}{\partial x} \Big|_E + \frac{\Delta x}{2} f|_{i,j} \end{aligned} \quad (10)$$

EDP relacionando C e E e substituindo (9) para ficar em função dos meios B e E

$$\begin{aligned} \frac{2}{\Delta x^2} \left[\mathbf{A}_{i,j+1} + \mathbf{A}_{i-1,j} - 2\mathbf{A}_{i,j} - \Delta x \frac{\partial A}{\partial y} \Big|_C + \Delta x \frac{\partial A}{\partial x} \Big|_E \right] &= f|_{i,j} \\ \frac{\mathbf{A}_{i,j+1} + \mathbf{A}_{i-1,j} - 2\mathbf{A}_{i,j}}{\Delta x} &= \frac{\partial A}{\partial y} \Big|_C - \frac{\partial A}{\partial x} \Big|_E + \frac{\Delta x}{2} f|_{i,j} \\ &= \frac{\mu_c}{\mu_b} \frac{\partial A}{\partial y} \Big|_B - \frac{\partial A}{\partial x} \Big|_E + \frac{\Delta x}{2} f|_{i,j} \end{aligned} \quad (11)$$

EDP relacionando B e D e substituindo (9) para ficar em função dos meios B e E

$$\begin{aligned} \frac{2}{\Delta x^2} \left[\mathbf{A}_{i,j-1} + \mathbf{A}_{i+1,j} - 2\mathbf{A}_{i,j} + \Delta x \frac{\partial A}{\partial y} \Big|_B - \Delta x \frac{\partial A}{\partial x} \Big|_D \right] &= f|_{i,j} \\ \frac{\mathbf{A}_{i,j-1} + \mathbf{A}_{i+1,j} - 2\mathbf{A}_{i,j}}{\Delta x} &= -\frac{\partial A}{\partial y} \Big|_B + \frac{\partial A}{\partial x} \Big|_D + \frac{\Delta x}{2} f|_{i,j} \\ &= -\frac{\partial A}{\partial y} \Big|_B + \frac{\mu_d}{\mu_e} \frac{\partial A}{\partial x} \Big|_E + \frac{\Delta x}{2} f|_{i,j} \end{aligned} \quad (12)$$

EDP relacionando C e D e substituindo (9) para ficar em função dos meios B e E

$$\begin{aligned} \frac{2}{\Delta x^2} \left[\mathbf{A}_{i,j+1} + \mathbf{A}_{i+1,j} - 2\mathbf{A}_{i,j} - \Delta x \frac{\partial A}{\partial y} \Big|_C - \Delta x \frac{\partial A}{\partial x} \Big|_D \right] &= f|_{i,j} \\ \frac{\mathbf{A}_{i,j+1} + \mathbf{A}_{i+1,j} - 2\mathbf{A}_{i,j}}{\Delta x} &= \frac{\partial A}{\partial y} \Big|_C + \frac{\partial A}{\partial y} \Big|_D + \frac{\Delta x}{2} f|_{i,j} \\ &= \frac{\mu_c}{\mu_b} \frac{\partial A}{\partial y} \Big|_B + \frac{\mu_d}{\mu_e} \frac{\partial A}{\partial y} \Big|_E + \frac{\Delta x}{2} f|_{i,j} \end{aligned} \quad (13)$$

Fazendo (11) - (10):

$$\begin{aligned} \left(\frac{\mu_c}{\mu_b} + 1 \right) \frac{\partial A}{\partial y} \Big|_B &= \frac{1}{\Delta x} [\mathbf{A}_{i,j+1} - \mathbf{A}_{i,j-1}] \\ \frac{\partial A}{\partial y} \Big|_B &= \frac{\mu_b}{\mu_c + \mu_b} \frac{1}{\Delta x} [\mathbf{A}_{i,j+1} - \mathbf{A}_{i,j-1}] \end{aligned} \quad (14)$$

Fazendo (12) - (10):

$$\begin{aligned} \left(\frac{\mu_d}{\mu_e} + 1 \right) \frac{\partial A}{\partial x} \Big|_E &= \frac{1}{\Delta x} [\mathbf{A}_{i+1,j} - \mathbf{A}_{i-1,j}] \\ \frac{\partial A}{\partial x} \Big|_E &= \frac{\mu_e}{\mu_d + \mu_e} \frac{1}{\Delta x} [\mathbf{A}_{i+1,j} - \mathbf{A}_{i-1,j}] \end{aligned} \quad (15)$$

Finalmente, substituindo (14) e (15) em (13):

$$\frac{\mathbf{A}_{i,j+1} + \mathbf{A}_{i+1,j} - 2\mathbf{A}_{i,j}}{\Delta x} = \frac{\mu_c}{\mu_c + \mu_b} \frac{1}{\Delta x} [\mathbf{A}_{i,j+1} - \mathbf{A}_{i,j-1}] + \frac{\mu_d}{\mu_d + \mu_e} \frac{1}{\Delta x} [\mathbf{A}_{i+1,j} - \mathbf{A}_{i-1,j}] + \frac{\Delta x}{2} f|_{i,j}$$

Pode-se isolar o termo de interesse $\mathbf{A}_{i,j}$ da seguinte forma:

$$\mathbf{A}_{i,j} = \frac{1}{2} \left\{ \mathbf{A}_{i,j+1} \left[\frac{\mu_b}{\mu_c + \mu_b} \right] + \mathbf{A}_{i,j-1} \left[\frac{\mu_c}{\mu_c + \mu_b} \right] + \mathbf{A}_{i+1,j} \left[\frac{\mu_e}{\mu_d + \mu_e} \right] + \mathbf{A}_{i-1,j} \left[\frac{\mu_d}{\mu_d + \mu_e} \right] - \frac{\Delta x^2}{2} f|_{i,j} \right\} \quad (16)$$

2.2.2 Regime Transiente

Com a inserção de um termo transiente no problema a EDP toma a seguinte forma:

$$\frac{\partial}{\partial x} \left(\frac{1}{\mu_y} \frac{\partial A}{\partial x} \right) + \frac{\partial}{\partial y} \left(\frac{1}{\mu_x} \frac{\partial A}{\partial y} \right) = \sigma \frac{\partial A}{\partial t} + J_z \cos(\omega t) \quad (17)$$

Além disso, sabe-se que as condições de contorno definidas na equação (9) não são alteradas. O enunciado declara, entretanto, que o valor das permeabilidades em x e em y não depende da posição, o que implica que $\frac{\partial(1/\mu_x)}{\partial x} = \frac{\partial(1/\mu_y)}{\partial y} = 0$, o que resulta na seguinte equação:

$$\frac{1}{\mu_y} \frac{\partial^2 A}{\partial x^2} + \frac{1}{\mu_x} \frac{\partial^2 A}{\partial y^2} = \sigma \frac{\partial A}{\partial t} + J_z \cos(\omega t) \quad (18)$$

Tomando $J_z \cos(\omega t) = f|_{i,j}^k$ e, novamente, aplicando o método proposto na Seção anterior, baseada no grid disposto na Figura 1, pode-se desenvolver uma equação geral válida para todo o domínio do problema que é capaz de resolver a EDP elíptica. A derivada temporal será aproximada pelo método da diferença central, as propriedades μ_x e μ_y são calculadas no ponto (i, j) , que é o ponto central da grid, e basta calcular a fórmula para o

cálculo das derivadas parciais em x e em y. Para tanto, aproveita-se as expansões em séries de Taylor anteriormente desenvolvidas:

EDP relacionando B e E:

$$\begin{aligned} \Delta x \left(\mu_x \frac{\partial A}{\partial x} \Big|_E^k + \mu_y \frac{\partial A}{\partial y} \Big|_B^k \right) + \mu_x A_{i-1,j}^k + \mu_y A_{i,j-1}^k - (\mu_x + \mu_y) A_{i,j}^k &= \underbrace{\frac{\sigma \mu_x \mu_y \Delta x^2}{2 \Delta t} [A_{i,j}^{k+1} - A_{i,j}^k]}_{\alpha} + \underbrace{\frac{\mu_x \mu_y \Delta x^2}{2} f \Big|_{i,j}^k}_{g|_{i,j}^k} \\ \Delta x \left(\mu_x \frac{\partial A}{\partial x} \Big|_E^k + \mu_y \frac{\partial A}{\partial y} \Big|_B^k \right) &= \alpha A_{i,j}^{k+1} - (\alpha - \mu_x - \mu_y) A_{i,j}^k - \mu_x A_{i-1,j}^k - \mu_y A_{i,j-1}^k + g|_{i,j}^k \end{aligned} \quad (19)$$

EDP relacionando C e E, substituindo (9):

$$\begin{aligned} \Delta x \left(\mu_x \frac{\partial A}{\partial x} \Big|_E^k - \mu_y \frac{\partial A}{\partial y} \Big|_C^k \right) + \mu_x A_{i-1,j}^k + \mu_y A_{i,j+1}^k - (\mu_x + \mu_y) A_{i,j}^k &= \alpha [A_{i,j}^{k+1} - A_{i,j}^k] + g|_{i,j}^k \\ \Delta x \left(\mu_x \frac{\partial A}{\partial x} \Big|_E^k - \mu_y \frac{\mu_c}{\mu_b} \frac{\partial A}{\partial y} \Big|_B^k \right) &= \alpha A_{i,j}^{k+1} - (\alpha - \mu_x - \mu_y) A_{i,j}^k - \mu_x A_{i-1,j}^k - \mu_y A_{i,j+1}^k + g|_{i,j}^k \end{aligned} \quad (20)$$

EDP relacionando B e D, substituindo (9):

$$\begin{aligned} \Delta x \left(-\mu_x \frac{\partial A}{\partial x} \Big|_D^k + \mu_y \frac{\partial A}{\partial y} \Big|_B^k \right) + \mu_x A_{i+1,j}^k + \mu_y A_{i,j-1}^k - (\mu_x + \mu_y) A_{i,j}^k &= \alpha [A_{i,j}^{k+1} - A_{i,j}^k] + g|_{i,j}^k \\ \Delta x \left(-\mu_x \frac{\mu_d}{\mu_e} \frac{\partial A}{\partial x} \Big|_E^k + \mu_y \frac{\partial A}{\partial y} \Big|_B^k \right) &= \alpha A_{i,j}^{k+1} - (\alpha - \mu_x - \mu_y) A_{i,j}^k - \mu_x A_{i+1,j}^k - \mu_y A_{i,j-1}^k + g|_{i,j}^k \end{aligned} \quad (21)$$

EDP relacionando C e D, substituindo (9):

$$\begin{aligned} \Delta x \left(-\mu_x \frac{\partial A}{\partial x} \Big|_D^k - \mu_y \frac{\partial A}{\partial y} \Big|_C^k \right) + \mu_x A_{i+1,j}^k + \mu_y A_{i,j+1}^k - (\mu_x + \mu_y) A_{i,j}^k &= \alpha [A_{i,j}^{k+1} - A_{i,j}^k] + g|_{i,j}^k \\ \Delta x \left(-\mu_x \frac{\mu_d}{\mu_e} \frac{\partial A}{\partial x} \Big|_E^k - \mu_y \frac{\mu_c}{\mu_b} \frac{\partial A}{\partial y} \Big|_B^k \right) &= \alpha A_{i,j}^{k+1} - (\alpha - \mu_x - \mu_y) A_{i,j}^k - \mu_x A_{i+1,j}^k - \mu_y A_{i,j+1}^k + g|_{i,j}^k \end{aligned} \quad (22)$$

Fazendo a manipulação algébrica (19) - (20):

$$\frac{\partial A}{\partial y} \Big|_B^k = \frac{\mu_b}{\mu_c + \mu_b} \frac{1}{\Delta x} [A_{i,j+1}^k - A_{i,j-1}^k] \quad (23)$$

Analogamente, com (19) - (21):

$$\frac{\partial A}{\partial x} \Big|_E^k = \frac{\mu_e}{\mu_e + \mu_d} \frac{1}{\Delta x} [A_{i+1,j}^k - A_{i-1,j}^k] \quad (24)$$

Finalmente, substituindo (23) e (24) em (22):

$$A_{i,j}^{k+1} = \frac{1}{\alpha} \left[\left(\frac{\mu_x \mu_e}{\mu_e + \mu_d} A_{i+1,j}^k + \frac{\mu_x \mu_d}{\mu_e + \mu_d} A_{i-1,j}^k + \frac{\mu_y \mu_b}{\mu_c + \mu_b} A_{i,j+1}^k + \frac{\mu_y \mu_c}{\mu_c + \mu_b} A_{i,j-1}^k \right) + (\alpha - \mu_x - \mu_y) A_{i,j}^k - g|_{i,j}^k \right] \quad (25)$$

Sendo as variáveis auxiliares:

$$\alpha = \frac{\sigma \mu_x \mu_y \Delta x^2}{2 \Delta t}$$

$$g|_{i,j}^k = \frac{\mu_x \mu_y \Delta x^2}{2} J_z \cos(\omega t)$$

No entanto, esta equação é definida para os pontos que pertencem dentro da bobina, em que o termo σ é definido e diferente de zero, desta forma pode-se isolar o termo $A_{i,j}^{k+1}$. Para os pontos fora deste domínio, o termo da variação temporal é multiplicado por 0, que é o valor de σ e então não se pode isolar o termo $A_{i,j}^{k+1}$, visto que $\alpha = 0$. Então, a equação para os termos fora da bobina é deduzida analogamente a equação (25) e isolando os termos $A_{i,j}^k$ e fazendo as simplificações $\alpha = 0$ e $g|_{i,j}^k = 0$, visto que que $J_z = 0$.

A equação deduzida, usando as condições descritas, para os pontos fora da bobina é

$$A_{i,j}^k = \frac{1}{\mu_x + \mu_y} \left(\frac{\mu_x \mu_e}{\mu_e + \mu_d} A_{i+1,j}^k + \frac{\mu_x \mu_d}{\mu_e + \mu_d} A_{i-1,j}^k + \frac{\mu_y \mu_b}{\mu_c + \mu_b} A_{i,j+1}^k + \frac{\mu_y \mu_c}{\mu_c + \mu_b} A_{i,j-1}^k \right) \quad (26)$$

2.3 Cálculo dos campos \mathbf{B} e \mathbf{H}

Com os métodos para definição do valor de \mathbf{A} estruturados para os diversos casos em toda malha, pode-se encontrar então o valor de \mathbf{B} através, novamente, do desenvolvimento das derivadas por diferenças centrais. Uma vez que se tem que \mathbf{B} é o rotacional de \mathbf{A} ($\mathbf{B} = \nabla \times \mathbf{A}$):

$$\mathbf{B} = \nabla \times \mathbf{A} \Rightarrow \mathbf{B} = \left(\frac{\partial A_z}{\partial y}, -\frac{\partial A_z}{\partial x} \right) \quad (27)$$

A partir da diferença central, têm-se então que

$$B_x|_{i,j} = \frac{\partial A_z}{\partial y}|_{i,j} = \frac{A_{i,j+1} - A_{i,j-1}}{2\Delta x} \quad (28)$$

$$B_y|_{i,j} = -\frac{\partial A_z}{\partial x}|_{i,j} = -\frac{A_{i+1,j} - A_{i-1,j}}{2\Delta x} \quad (29)$$

Assim, para encontrar o valor do campo vetorial \mathbf{B} em toda a malha basta iterar i, j e calcular os valores de B_x e B_y para cada ponto e armazenar estes valores em duas matrizes com mesma dimensão de \mathbf{A} .

O cálculo de \mathbf{H} é extremamente simples: para cada ponto na malha deve-se dividir as componentes de \mathbf{B} pelos valores de μ_x e μ_y :

$$H_x|_{i,j} = \frac{B_x}{\mu_x}|_{i,j} \quad (30)$$

$$H_y|_{i,j} = \frac{B_y}{\mu_y}|_{i,j} \quad (31)$$

2.4 Cálculo da força eletromagnética na armadura

A força eletromagnética na armadura tem duas componentes, que podem ser encontradas através das seguintes integrais de linha no contorno externo da armadura:

$$\begin{aligned} F_{ela,x} &= \frac{1}{2\mu_0} \oint_{\Gamma_{\text{armadura}}} [(B_x^2 - B_y^2) \cdot n_x + 2B_x B_y \cdot n_y] dl \\ F_{ela,y} &= \frac{1}{2\mu_0} \oint_{\Gamma_{\text{armadura}}} [2B_x B_y \cdot n_x + (B_y^2 - B_x^2) \cdot n_y] dl \end{aligned} \quad (32)$$

O contorno externo Γ_{armadura} é um retângulo regular que abrange o espaço $[0, 0.04] \times [-0.1, 0.1]$. Para fins de resolução desta integral, pode-se dividir este caminho em 4 partes: a primeira, em $y = -0.1$, com x variando em $[0, 0.04]$; a segunda, em $x = 0.04$, com y variando em $[-0.1, 0.1]$; a terceira, em $y = 0.1$, com x variando de $[0.04, 0]$; e, por fim, a quarta, em $x = 0$, com y variando em $[0.1, -0.1]$.

Por conta da condição de contorno encontrada em (6) para o contorno externo da malha, sabe-se que \mathbf{B} deve ser **nulo** para os 3 últimos trechos, uma vez que ambas as derivadas de A_z , em x e em y para todos os pontos da fronteira externa são nulos. Com isto, torna-se necessário analisar apenas o comportamento do campo no primeiro trecho, onde $\mathbf{n} = e_x$:

$$\begin{aligned} F_{\text{ela},x} &= \frac{1}{2\mu_0} \int_{-0.1}^{0.1} \left(B_x^2 \Big|_{x=0.04} - B_y^2 \Big|_{x=0.04} \right) dy \\ F_{\text{ela},y} &= \frac{1}{2\mu_0} \int_{-0.1}^{0.1} \left(2B_x \Big|_{x=0.04} B_y \Big|_{x=0.04} \right) dy \end{aligned} \quad (33)$$

Contudo, no código-fonte A, usou-se o método dos trapézios para o cálculo desta integral numericamente. Como o método foi desenvolvido de forma a ser versátil, considerou-se os demais termos na formulação, que são zerados devido às condições 6, para caso fosse desejado o cálculo da força em outras condições.

3 Itens a), b), c), e d)

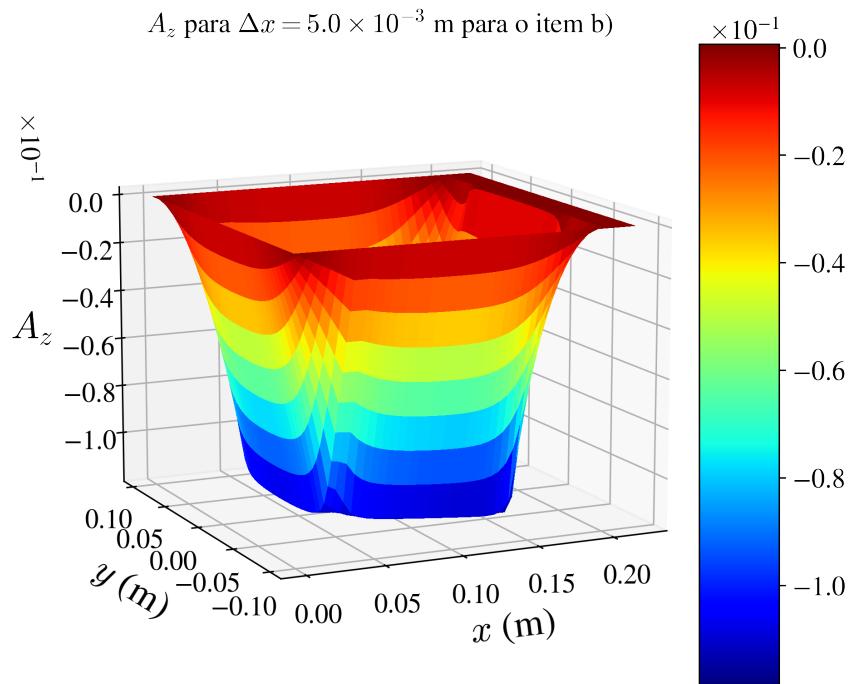
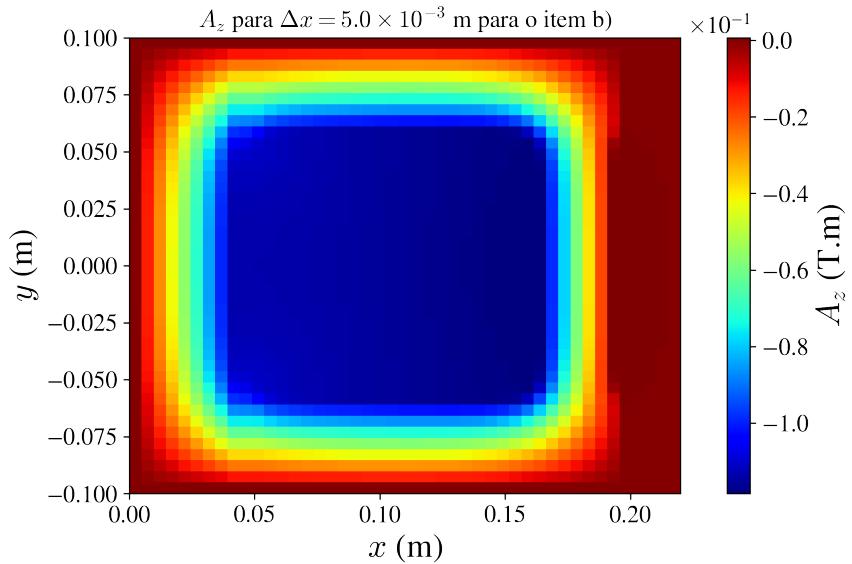
Para os itens a), b), c), e d) foi realizada a solução da equação (3) utilizando o Método das Diferenças Finitas (MDF) com uma malha quadrada ($\Delta x = \Delta y$), por meio da modelagem e equacionamento desenvolvidos na Seção 2.2.1. Foi implementado o método da “sobrerrelaxação” com um valor de $\lambda = 1.75$. Para a escolha dos três valores de Δx , tomou-se como base conhecimentos prévios no tema e procurou-se discretizar de maneira coerente baseada abertura do entreferro entre a armadura e o núcleo de ferro, que tem 1 cm, portanto os valores de Δx escolhidos levaram em consideração esse tamanho e o número de pontos que se desejou ter na grid para este meio. Como valor de discretização grande, escolheu-se $\Delta x = 5 \cdot 10^{-3}$ m, que equivale a 2 pontos no entreferro, na direção x . Como valor intermediário de discretização, adotou-se $\Delta x = 2.5 \cdot 10^{-3}$ m, que representa o dobro do valor anterior de pontos no meio de interesse. Finalmente, o último valor de Δx foi de $1 \cdot 10^{-3}$ m, que representa 10 pontos na direção no eixo x para o entreferro, que já gerou uma alta demanda computacional.

3.1 $\Delta x = 5 \cdot 10^{-3}$ m

A primeira discretização tomada, considerada grande, foi a de $5 \cdot 10^{-3}$ m.

3.1.1 Análise do potencial eletromagnético A_z

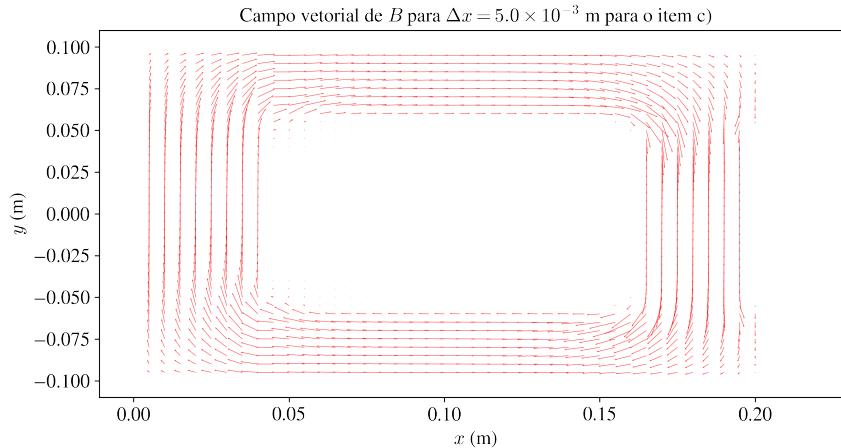
Os valores para o potencial magnético (A_z) encontrados podem ser visualizados nas Figuras 2 e 3

Figura 2: Distribuição de A_z para $\Delta x = 5 \cdot 10^{-3}$ mFigura 3: Perfil de A_z para $\Delta x = 5 \cdot 10^{-3}$ m

Percebe-se que com a discretização da malha quadrada de forma grande, o potencial percorre o núcleo de ferro, passa pelo entreferro, e chega até a armadura mantendo a mesma distribuição com as interfaces externas com menor potencial, em módulo, comparados com a interface com o ar. A parte interna, cujo material é o ar, possui o menor potencial e é quase constante no meio todo, sendo quase plana essa parte na Figura 2.

3.1.2 Análise do campo vetorial B

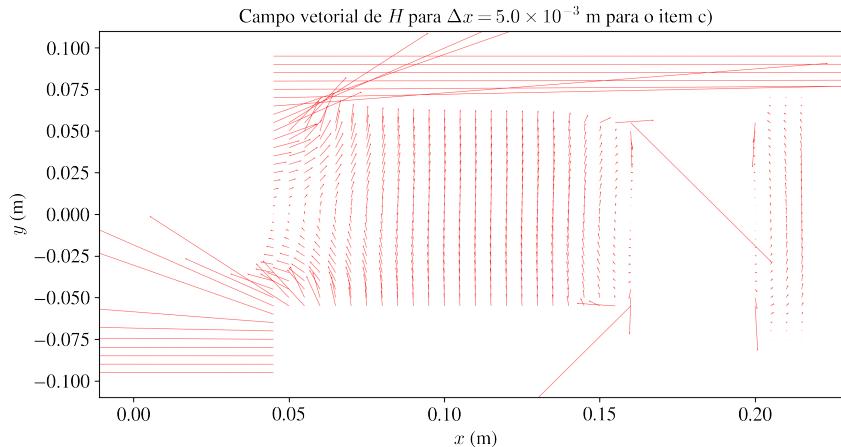
O resultado encontrado para o vetor **B** no caso da discretização mais grosseira é apresentado na Figura 4

Figura 4: Campo vetorial \mathbf{B} para $\Delta x = 5 \cdot 10^{-3}$ m

Nele, observa-se que o vetor de densidade superficial de fluxo magnético (\mathbf{B}) é maior no trecho que envolve o núcleo de ferro, entreferro e armadura. De fato, pelo efeito da forçante da bobina (J_z) percebe-se que este campo teria de apresentar maiores módulos justamente nestes meios. Ademais, pela definição, $\mathbf{B} = \nabla \times \mathbf{A}$, o campo \mathbf{B} deve ser maior nos trechos onde há um maior declive na distribuição de \mathbf{A} , devido a isto a parte quase plana que representa o ar indica um valor de \mathbf{B} menor, encontrado no centro da Figura 4.

3.1.3 Análise do campo vetorial \mathbf{H}

O vetor de intensidade de campo magnético (\mathbf{H}) encontrado para o caso com a discretização grosseira pode ser visto na Figura 5

Figura 5: Campo vetorial \mathbf{H} para $\Delta x = 5 \cdot 10^{-3}$ m

Ao se observar o campo vetorial de \mathbf{H} , percebe-se que apesar da discretização grosseira um dos princípios físicos que rege o sistema é amplificado, o fato da energia ser concentrada no entreferro. Como comentado, na discretização com Δx grande, escolheu-se de forma a ter ao menos 2 pontos no eixo x que representassem o entreferro, já que esta parte que deve concentrar a energia. Como há uma grande concentração de energia em poucos pontos, o valor de H é amplificado neste trecho. Além disso, percebe-se que devido a permeabilidade magnética do ar ser relativamente maior do que a do ferro, o vetor \mathbf{H} é praticamente expressivo apenas nos meios cuja permeabilidade é igual ao do ar, ou seja, a bobina e o próprio ar. Além disso, percebe-se um outro efeito que é ao lado dos pontos de virada do núcleo de ferro, ali o campo se comporta como se fosse um entreferro e também há energia concentrada lá, como pode ser visto pelo módulo do vetor com origem neste ponto.

3.1.4 Análise da força eletromagnética F_{ela}

O cálculo da força eletromagnética, das componentes $F_{ela,x}$ e $F_{ela,y}$, para a discretização grosseira, obtém-se um resultado não coerente com os valores reais esperados.

$$1. F_{ela,x} = 37\,089.02 \text{ N}$$

$$2. F_{ela,y} = -845.80 \text{ N}$$

Após pesquisa de valores reais de *maglev*, sabe-se que a ordem destas forças deve ser $1 \cdot 10^2 \text{ N}$. Além disso, a componente em y deve ser, idealmente, nula, visto que há simetria no eixo y . Essa discrepância de dá justamente pela aproximação grosseira do método ($\Delta x = 5 \cdot 10^{-3} \text{ m}$), que gerou valores não reais para os vetores \mathbf{B} e \mathbf{H} , o que acarretou em valores não próximos ao esperado.

$$3.2 \quad \Delta x = 2.5 \cdot 10^{-3} \text{ m}$$

A discretização mediana adotada foi a de $2.5 \cdot 10^{-3} \text{ m}$.

3.2.1 Análise do potencial eletromagnético A_z

Os resultados do potencial magnético (A_z) encontrados para a discretização mediana podem ser visualizados nas Figuras 6 e 7

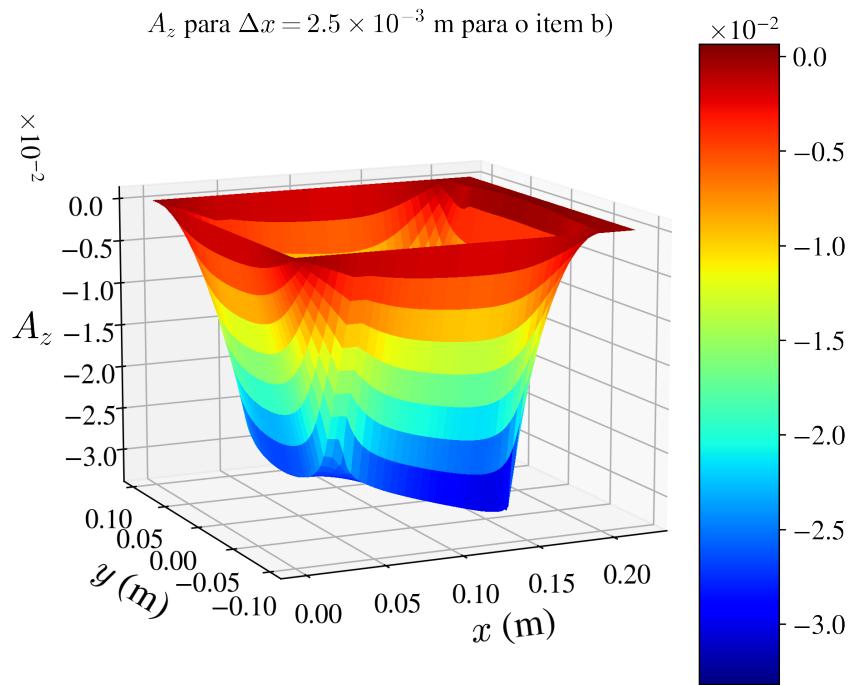
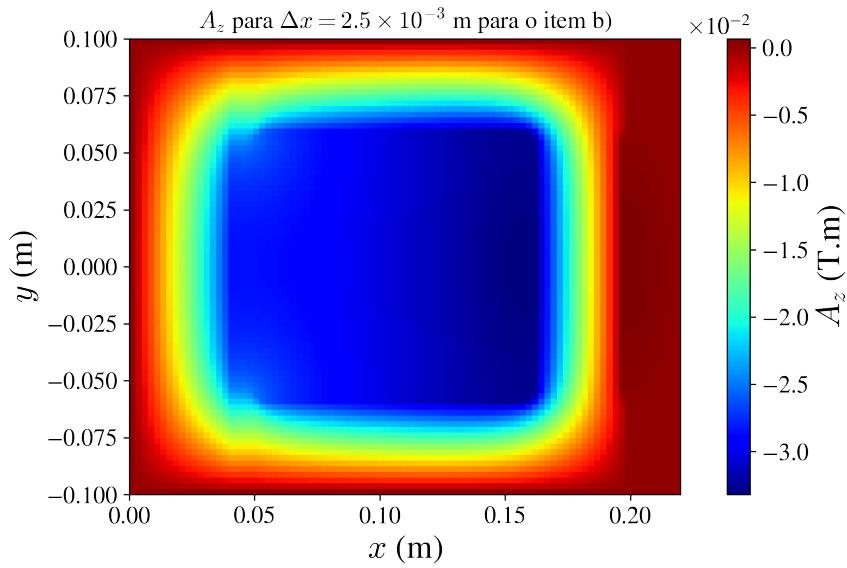


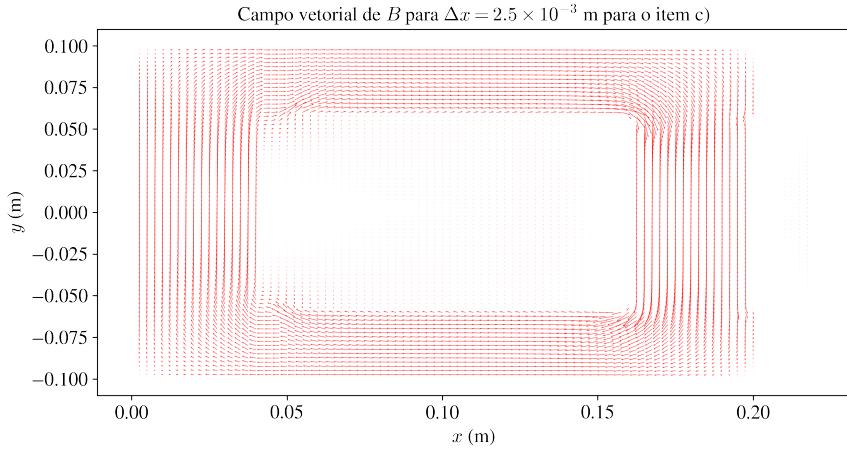
Figura 6: Distribuição de A_z para $\Delta x = 2.5 \cdot 10^{-3} \text{ m}$

Figura 7: Perfil de A_z para $\Delta x = 2.5 \cdot 10^{-3}$ m

Após a análise da distribuição e perfil, para a discretização mediana, percebe-se que novamente a distribuição dentro do núcleo de ferro, entreferro, e armadura, permanece essencialmente a mesma, com um gradiente indo das faces externas para as internas. Novamente dentro do ar percebe-se pouca variação do potencial magnético, mas desta vez não é um plano quase perpendicular ao plano xy . Com isso, tem-se um gradiente relativamente distinto, como será discutido na próxima subseção. Com a discretização maior comparada ao item anterior, percebe-se que o gráfico tem o perfil apresentando menos desniveis bruscos.

3.2.2 Análise do campo vetorial **B**

Os valores encontrados das componentes do vetor de densidade superficial de fluxo magnético **B** para a discretização intermediária são mostrados na Figura 8

Figura 8: Campo vetorial **B** para $\Delta x = 2.5 \cdot 10^{-3}$ m

Para o vetor **B**, percebe-se novamente a mesma tendência de distribuição concentrada nos materiais ferrosos (núcleo e armadura) e no entreferro. Nesta discretização, percebe-se um pequeno gradiente em direção a armadura, que é evidência da pequena inclinação encontrada no gráfico 6 no meio do ar. Percebe-se também que a condição de contorno 8 é crucial para a determinação do valor de **B** nos meios, já que há uma diferença entre os valores de μ da ordem de 2500 vezes.

3.2.3 Análise do campo vetorial \mathbf{H}

O vetor de intensidade de campo magnético \mathbf{H} no caso da discretização mediana escolhida é representado na Figura 9

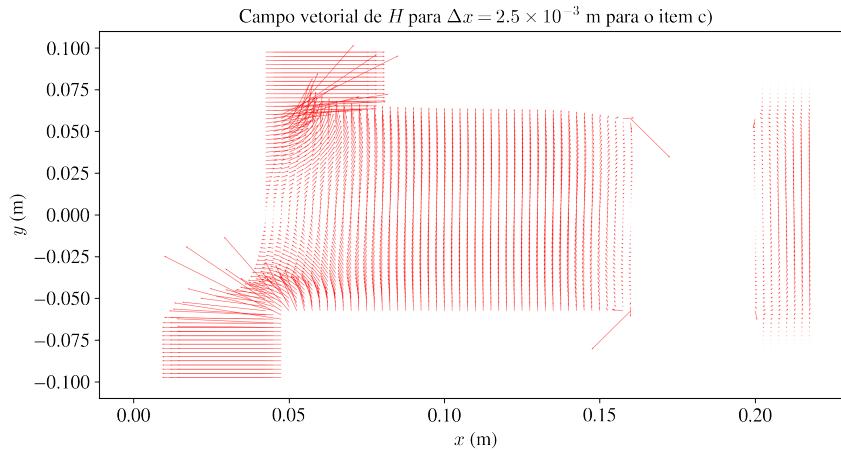


Figura 9: Campo vetorial \mathbf{H} para $\Delta x = 2.5 \cdot 10^{-3}$ m

Neste gráfico, percebe-se novamente um dos fenômenos extensamente estudados para o projeto de motores e também de outros sistemas magnéticos: a concentração da energia de campo magnético no entreferro. Isso se dá pela diferença entre as permeabilidades magnéticas e pela densidade superficial de fluxo magnético ser um ciclo dos materiais ferrosos. Isso implica que com a passagem de fluxos desta ordem no entreferro (ar) o valor da intensidade do campo seja cerca de 2500 vezes maior no ar. A imagem mostra claramente que dentro do ar tem-se uma intensidade razoável e no entreferro tem-se a concentração da intensidade, quando comparadas a qualquer um dos materiais cuja permeabilidade seja a do Ferro.

3.2.4 Análise da força eletromagnética F_{ela}

O cálculo da força eletromagnética, das componentes $F_{ela,x}$ e $F_{ela,y}$, para a discretização intermediária, obtém-se um resultado melhor do que o pior valor de discretização.

1. $F_{ela,x} = 2220.63$ N

2. $F_{ela,y} = -10.04$ N

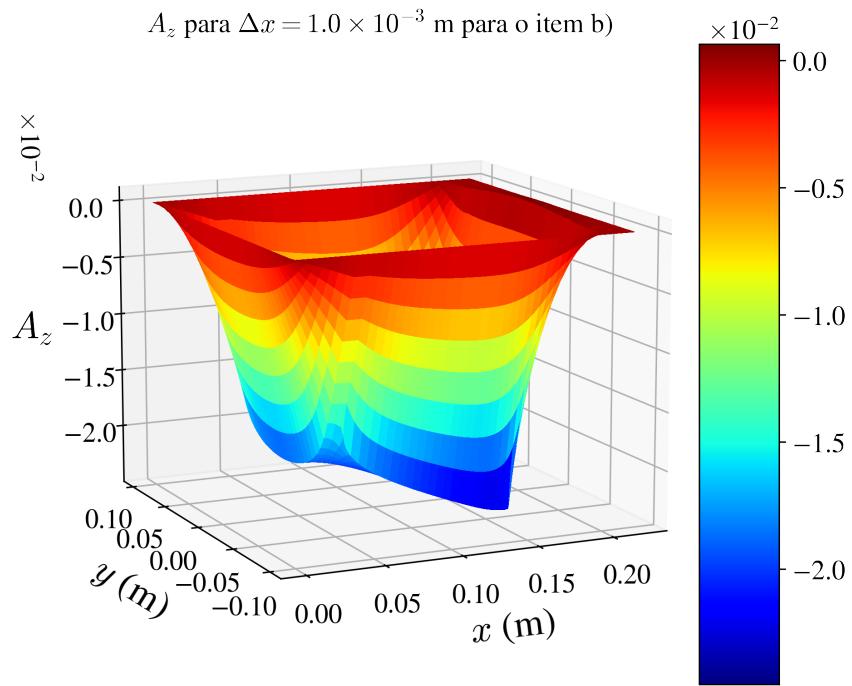
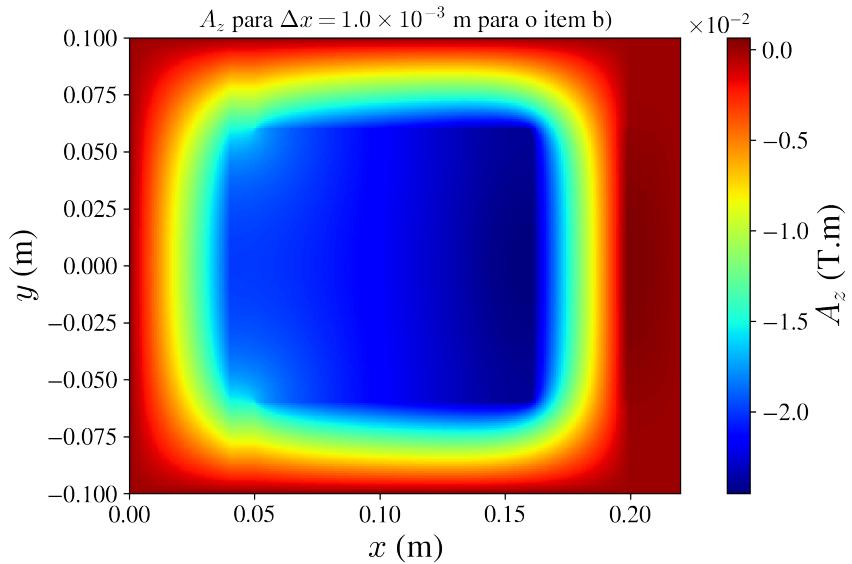
Apesar da melhora frente a outra discretização, percebe-se que ainda a componente $F_{ela,x}$ não está da ordem de grandeza esperada. Contudo, também percebe-se que o valor da componente $F_{ela,y}$ é muito mais próximo ao valor esperado, que é 0.

3.3 $\Delta x = 1 \cdot 10^{-3}$ m

Para a discretização pequena, tomou-se o valor de $1 \cdot 10^{-3}$ m.

3.3.1 Análise do potencial eletromagnético A_z

A componente em z do vetor potencial eletromagnético (\mathbf{A}) calculada no caso da maior discretização ($\Delta x = 1 \cdot 10^{-3}$ m)

Figura 10: Distribuição de A_z para $\Delta x = 1 \cdot 10^{-3}$ mFigura 11: Perfil de A_z para $\Delta x = 1 \cdot 10^{-3}$ m

Analisando os gráficos gerados, tanto o de distribuição quanto o de perfil, que a concentração do potencial eletromagnético é maior (em módulo) nos arredores da bobina e esta se dissipar ao se afastar da bobina. Isso ocorre em decorrência da maior discretização, que é coerente com experimentos reais deste fenômeno, já que há a dissipação da energia pelas transições dentro do meio. Com esta discretização maior, percebe-se claramente a diferença entre meios indicando uma mudança na inclinação ao se mudar de meio, de forma bem definida. Além disso, percebe-se que com a solução espacial há uma grande dependência da forçante J_z no formato geral do potencial eletromagnético, que é moldado de forma similar a uma cossenoide.

3.3.2 Análise do campo vetorial **B**

Os valores encontrados para as componentes do vetor **B** para a maior discretização, podem ser vistos no gráfico do campo vetorial na Figura 12

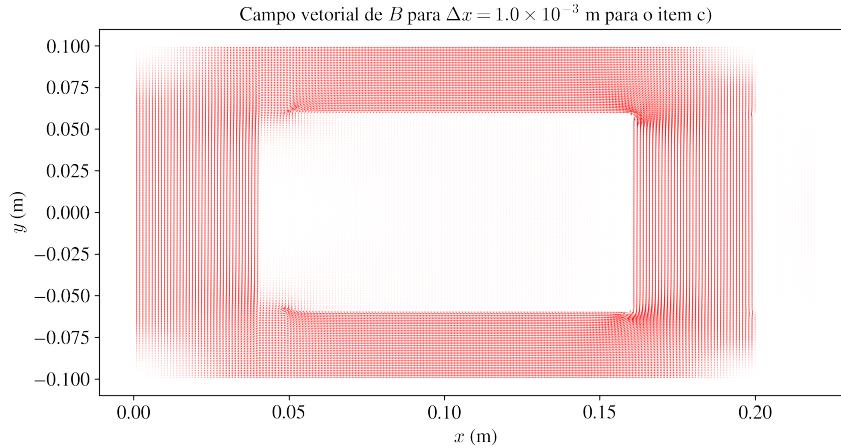


Figura 12: Campo vetorial **B** para $\Delta x = 1 \cdot 10^{-3}$ m

Novamente pode-se analisar o vetor de densidade superficial de fluxo magnético (**B**), para o caso da maior discretização e analisar o ciclo que se forma por esse fluxo ser induzido nos materiais ferrosos e no entreferro, que é em sentido horário na figura em análise. Este valor é coerente com o conhecido, como a regra da mão direita. Além disso, este vetor tem a direção aparentemente paralela ao segmento que ele está, isto é, nos trechos em x ele tem sua maior componente sendo B_x e B_y sendo quase nula, e analogamente para os trechos em y.

3.3.3 Análise do campo vetorial **H**

O vetor de intensidade de campo magnético **H** no caso da discretização mediana escolhida é representado na Figura 9

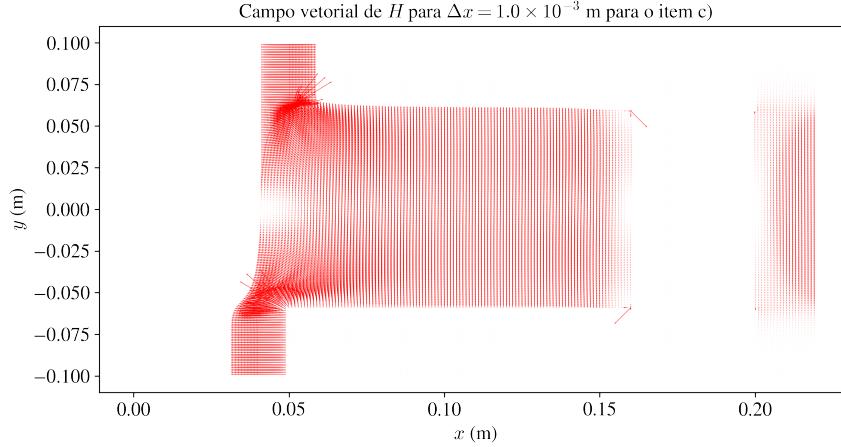


Figura 13: Campo vetorial **H** para $\Delta x = 1 \cdot 10^{-3}$ m

Para a análise do caso mais discretizado, inicialmente ressalta aos olhos o fato dos maiores valores de **H** serem no entreferro. De fato, como já foi extensamente discutido, é onde há a concentração da energia do campo magnético. Comparando-o com a Figura 4, percebe-se que o “entreferro” das quinas é novamente ressaltado no gráfico, como pode-se ver os vetores com origem nestes pontos. Além disso, percebe-se visualmente que dentro do entreferro o campo se alinha a direção do fluxo magnético

3.3.4 Análise da força eletromagnética F_{ela}

O cálculo da força eletromagnética, das componentes $F_{ela,x}$ e $F_{ela,y}$, para a maior discretização, obtém-se valores mais próximos aos esperados.

1. $F_{ela,x} = 1107.68 \text{ N}$
2. $F_{ela,y} = -0.74 \text{ N}$

O valor da componente em x da força eletromagnética está se aproximando do esperado (da ordem de $1 \cdot 10^2 \text{ N}$), mas com o refinamento da malha ela mudou drasticamente. Já com relação a componente em y , percebe-se que está claramente convergindo para o valor esperado (0 N) com o refinamento da malha. Então, apesar de ainda não ter se estabilizado percebe-se que há uma convergência com refinamento da malha, mesmo que lenta, para os valores esperados.

3.4 Análise da influência da discretização

Após a análise de cada uma das variáveis de interesse para os 3 casos de discretização ($\Delta x = 5 \cdot 10^{-3} \text{ m}$, $\Delta x = 2.5 \cdot 10^{-3} \text{ m}$, $\Delta x = 1 \cdot 10^{-3} \text{ m}$), e a comparação com os resultados esperados de *mageus* conhecidos, pode-se afirmar que quanto maior a discretização, mais acurados são os resultados encontrados, chegando a diferir por ordens de grandeza caso se tenha um valor inadequado de Δx . Percebe-se que o formato geral da superfície de A_z plotada nas Figuras 10, 6, 2 é respeitado, mas as ordens de grandeza não. Além disso, ao se comparar os valores de força, percebe-se que não há uma estabilidade ainda (pois da discretização intermediária para a melhor discretização há uma redução brusca de 50% na componente $F_{ela,x}$).

Além disso, é válido ressaltar que a modelagem abordada, vide a Figura 1, há uma aproximação que comete um erro geométrico (já que assume que na realidade todos os pontos são fronteira entre os pontos ao seu redor e desta forma não há fronteiras bem definidas como poderia se ter em uma formulação alternativa, onde deveria se recair em casos para a análise da formulação - como fronteiras verticais e horizontais).

Desta forma, pode-se afirmar que o método desenvolvido para a resolução do problema posto por métodos de diferenças finitas, é altamente dependente de valores adequados de discretização da malha quadrada. Os valores escolhidos, em especial o mais refinado ($\Delta x = 1 \cdot 10^{-3} \text{ m}$), têm uma grande demanda computacional (acredita-se que em decorrência do software de escolha, *Python*, que por ser uma linguagem muti-propósito não é a mais adequada para implementação do método das diferenças finitas)

Ademais, a resolução do problema através da formulação proposta (ou seja, tomando todos os pontos da malha com interfaces quadruplicas entre 4 outros meios) permite desenvolver uma fórmula geral da sobrerelaxação para todo o domínio analisado. O sistema pode, então, ser desenvolvido em uma forma matricial com a matriz de coeficientes sendo uma matriz quadrada e esparsa de dimensão $(M - 1) \times (N - 1) \times (M - 1) \times (N - 1)$, a matriz de incógnitas sendo uma matriz-coluna de dimensão $(M - 1) \times (N - 1) \times 1$ contendo os valores de A_z organizados em ordem lexicográfica e a matriz a direita sendo uma matriz-coluna de igual dimensão contendo as condições de contorno e valores da forçante (corrente). Esta montagem proposta poderia incrementar em muito a eficiência do sistema, uma vez que cada iteração da sobrerelaxação poderia ser resolvida em um passo através dos resolutores de sistemas lineares especializados para matrizes esparsas. Este ganho de eficiência abrir caminho para a implementação de discretizações significantemente menores, o que consequentemente garantiria resultados mais fieis à realidade quanto possível.

4 Item e.1)

Para o caso deste exercício, têm-se um material não isotrópico, mas as demais condições são iguais as do exercício anterior. Para este caso escolheu-se o valor de $\Delta x = 2.5 \cdot 10^{-3} \text{ m}$, que possuía uma demanda computacional razoável e resultados próximos ao esperado nos casos anteriores

4.1 Análise do potencial eletromagnético A_z

Os valores de A_z do vetor potencial eletromagnético (**A**) para o material não isotrópico sem regime transiente.

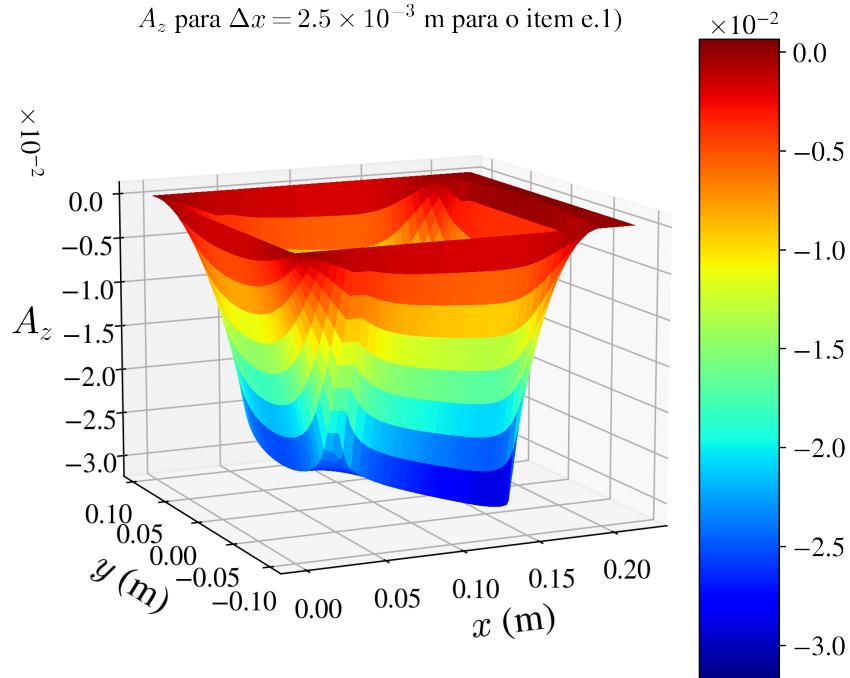


Figura 14: Distribuição de A_z para material não isotrópico e sem regime transiente

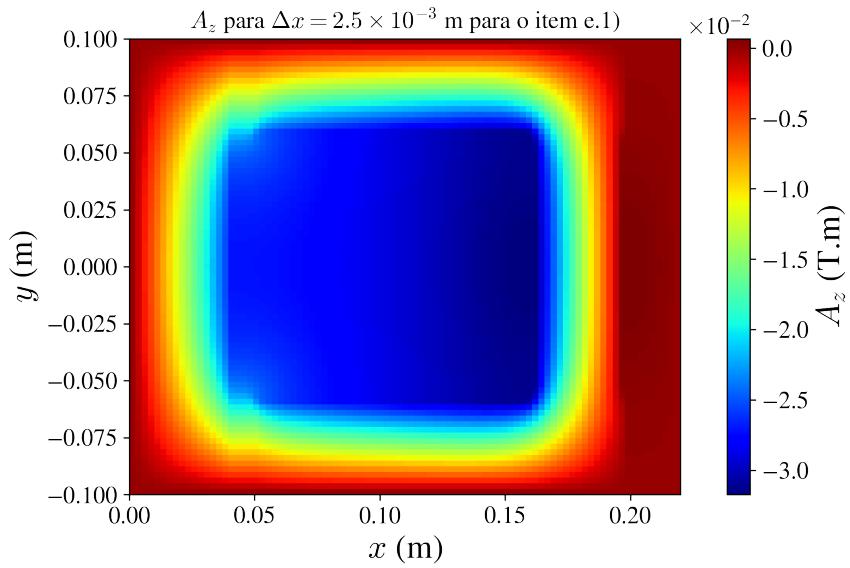


Figura 15: Perfil de A_z para material não isotrópico e sem regime transiente

Para este caso estudado, percebe-se novamente que o perfil e a distribuição se mantêm semelhantes a de outros valores encontrados. Isso se dá pois usou-se o mesmo método da sobrerelaxação para gerar estes resultados. Essencialmente a diferença esperada é que a permeabilidade do meio seja diferente frente ao caso com mesma discretização e com material isotrópico (vide Figura 7). De fato, ao se comparar ambas imagens percebe-se uma pequena diferença nas interfaces entre meios, visto que é onde há maior impacto da diferença da permeabilidade.

Percebe-se que com esta discretização os valores encontrados são iguais a esta discretização no caso isotrópico.

4.2 Análise do campo vetorial **B**

Os valores encontrados para as componentes do vetor **B** para este caso estudado podem ser vistos na Figura ??

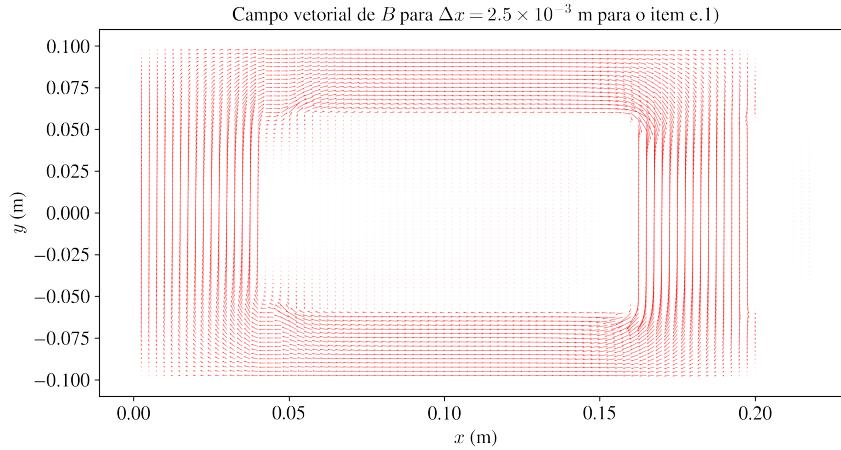


Figura 16: Campo vetorial **B** para $\Delta x = 1 \cdot 10^{-3}$ m

Mais uma vez, percebe-se a semelhança entre este resultado e o da simulação com mesma discretização e com material isotrópico, apresentado na Figura 8. O vetor densidade superficial de fluxo magnético possui o mesmo formato, com os mesmos pontos ressaltados. A diferença entre as imagens se da no entreferro, já que é a interface na qual há uma diferenciação pelo material ser não isotrópico e desta forma a passagem de fluxo é alterada.

4.3 Análise do campo vetorial **H**

O vetor de intensidade de campo magnético **H** pode ser visto na Figura 17

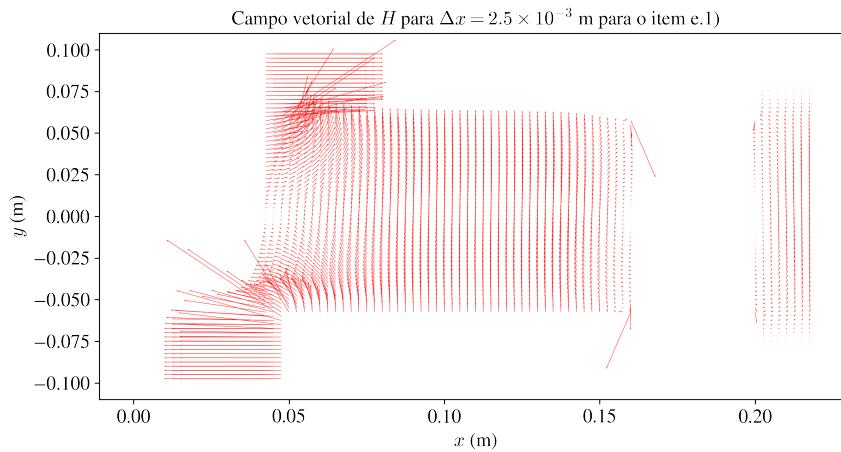


Figura 17: Campo vetorial **H** para $\Delta x = 1 \cdot 10^{-3}$ m

Para a análise deste caso, em especial comparando-o com o mesmo caso de discretização mas isotrópico, na Figura 8. Percebe-se que dentro dos materiais ferrosos o comportamento do vetor é, essencialmente, o mesmo, respeitando inclusive as magnitudes. A anisotropia do material é evidenciada pelo vetor com origem nas “quinas”, que corresponde a um ponto de características semelhantes a do entreferro. Nestes pontos, percebe-se que a

inclinação do vetor não é mais de 45° , que corresponderia a material com $\mu_x = \mu_y$, e passa-se a ter uma inclinação maior na direção y , já que este valor de permeabilidade é maior.

4.4 Análise da força eletromagnética F_{ela}

Com cálculo da força eletromagnética, das componentes $F_{\text{ela},x}$ e $F_{\text{ela},y}$, no modelo não isotrópico estudado, obtém-se os seguintes valores

1. $F_{\text{ela},x} = 84.91 \text{ N}$
2. $F_{\text{ela},y} = 1.06 \text{ N}$

Os valores das componentes da força são próximos ao esperado, ao menos a ordem de grandeza se encaixa nos valores pesquisados. Isso se dá devido ao bom refinamento da malha, e pelo fato do material apresentar ruído nos cálculos na componente B_x que é mitigado pelo fato da permeabilidade que o pondera ser menor frente a em y , ou seja, os erros do método computacional são reduzidos e gera-se um valor mais coerente de força. Este modelo não isotrópico faz com que a não se perca muita força eletromagnética na direção desejada frente ao cenário ideal de material isotrópico.

5 Item e.2)

Para este item, de maneira análoga ao item anterior, 4, usou-se o valor de $\Delta x = 2.5 \cdot 10^{-3} \text{ m}$.

5.1 Escolha de Δt

A escolha do valor de Δt para o regime transitório deve ser tomada a partir da análise de estabilidade do método explícito.

Sabe-se que o método explícito em duas direções é estável se, e somente se,

$$\Delta t \leq \frac{1}{8} \frac{(\Delta x)^2 + (\Delta y)^2}{k} \quad (34)$$

Com k sendo o termo que multiplica os termos da derivada de segunda ordem de A_z . Além disso, por se tratar de uma malha quadrada, $\Delta x = \Delta y$. Neste caso, têm-se duas constantes multiplicando estes termos. Para garantir a estabilidade, devemos então tomar por limite superior Δt o menor valor obtido a partir destes dois valores de k , sendo eles $k_1 = \frac{1}{\sigma \mu_{x,\text{bobina}}}$ e $k_2 = \frac{1}{\sigma \mu_{y,\text{bobina}}}$:

$$\Delta t \leq \frac{\sigma}{4} (\Delta x)^2 \min \{\mu_x, \mu_y\} \quad (35)$$

Como a bobina possui a mesma permeabilidade magnética que o ar, e esta não varia conforme a direção, chega-se a seguinte relação:

$$\Delta t \leq \frac{\sigma \mu_{\text{Ar}}}{4} (\Delta x)^2 \quad (36)$$

Encontrado o limite superior, tomou-se um valor de Δt 4× menor para se ter segurança da estabilidade. Assim:

$$\Delta t = \frac{\sigma \mu_{\text{Ar}}}{16} (\Delta x)^2 \quad (37)$$

5.2 Análise do potencial eletromagnético A_z

O potencial eletromagnético A_z no instante de tempo $t = 500\Delta t$ pode ser visto na Figura 18

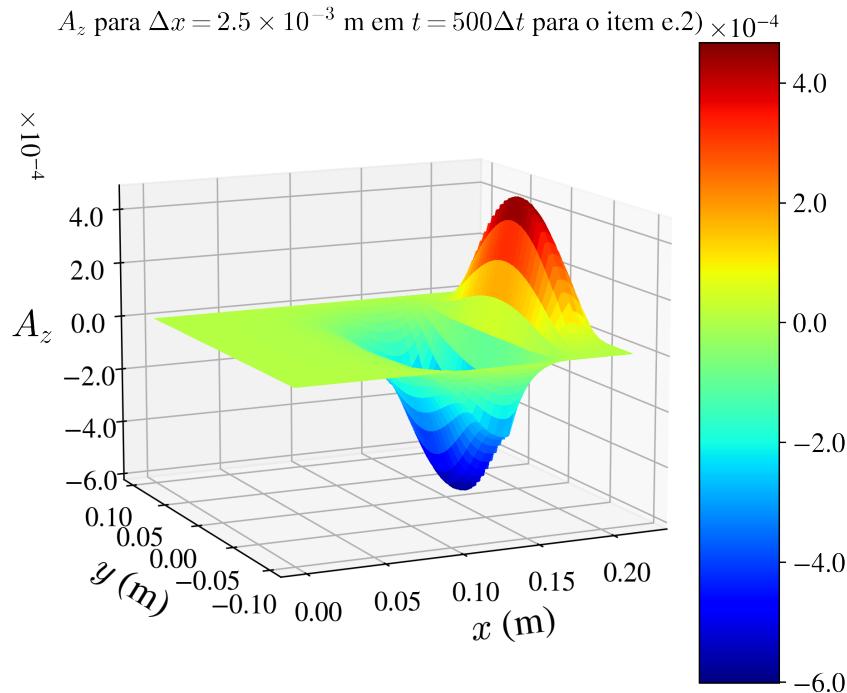
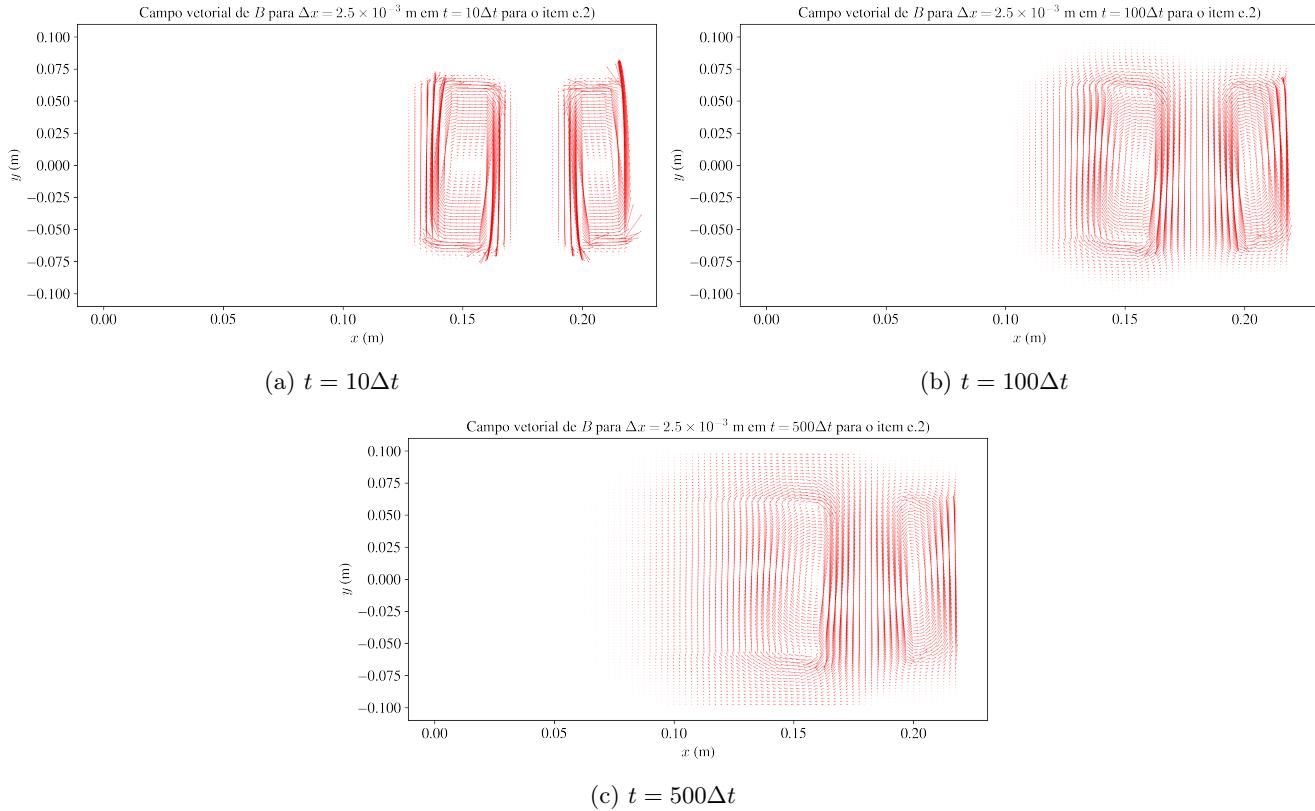


Figura 18: Distribuição de A_z para material não isotrópico e em regime transiente

Percebe-se que não houve tempo o suficiente para que o potencial eletromagnético se propague pelo meio. Há, claramente, um grande potencial na bobina devido à corrente imposta. Percebe-se que o sistema caminha para uma configuração semelhante à encontrada nos outros métodos. Pelo fato de não haver sobrerelaxação e o método iterar menos vezes (uma para cada instante de tempo), a ordem de grandeza dos valores encontrados não é igual a dos outros resultados.

5.3 Análise do campo vetorial \mathbf{B}

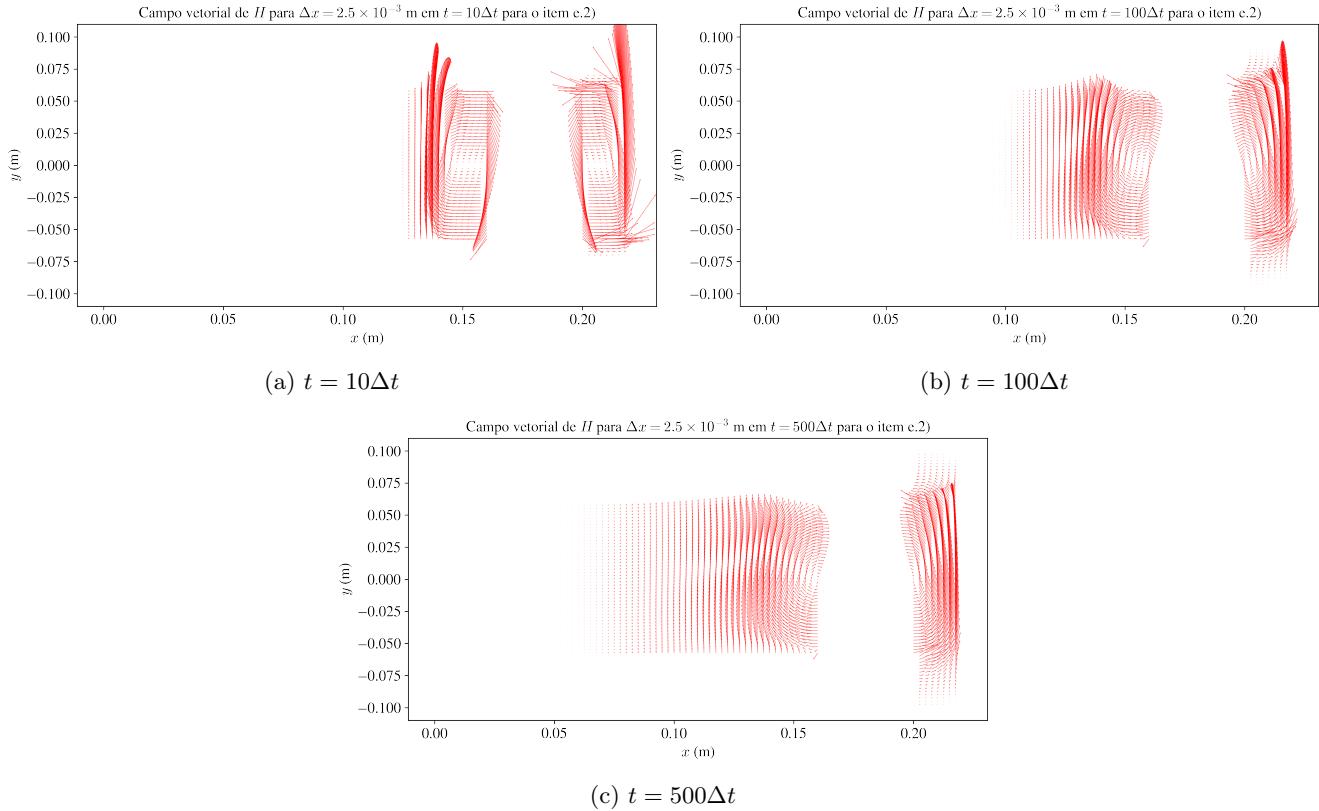
A progressão temporal do vetor \mathbf{B} nos instantes de tempo selecionados pode ser vista na Figura 19.

Figura 19: Campo vetorial **B** para os instantes de interesse

Com a progressão do vetor **B** no tempo, percebe-se a tendência de homogenização do vetor. Além disso, vê-se a velocidade de propagação do vetor de densidade superficial de fluxo magnético nos materiais. Percebe-se que inicialmente os arredores da bobina que possuem uma maior densidade superficial de fluxo magnético, visto que é devido a forçante gerada na bobina (J_z) que se inicia a propagação.

5.4 Análise do campo vetorial **H**

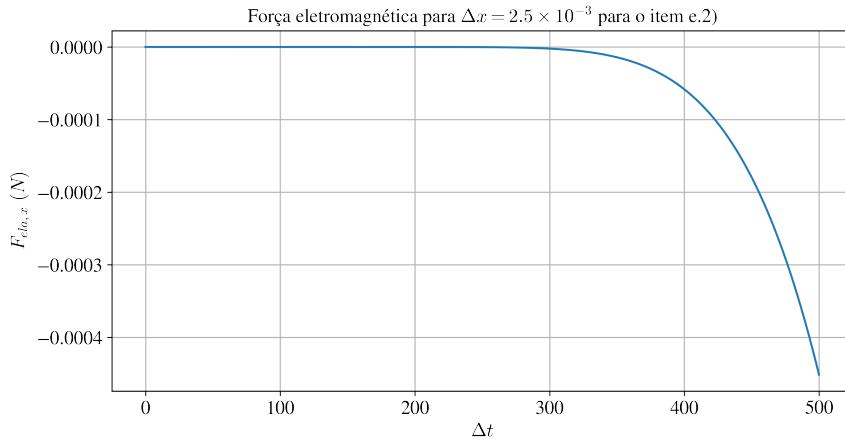
Os valores do vetor **H** são mostrados nos instantes de tempo solicitados na Figura 20

Figura 20: Campo vetorial \mathbf{H} para os instantes de interesse

Com os valores do vetor \mathbf{H} , percebe-se a tendência de homogenização do vetor de intensidade de campo magnético. Percebe-se que a propagação é quase nula nos materiais ferrosos, independente do instante de tempo, que está de acordo com o encontrado no regime permanente nas outras simulações.

5.5 Análise da força eletromagnética $F_{ela,x}$

A Figura 21 mostra a evolução temporal da intensidade da força eletromagnética na direção x com o avanço do tempo entre o instante inicial e $t = 500\Delta t$. Percebe-se que a força inicia em 0, como era de se esperar, uma vez que para o instante inicial, \mathbf{A} é a matriz nula e, portanto, todas as suas derivadas, que por sua vez definem o campo \mathbf{B} que é utilizado para o cálculo da força, também são nulas.

Figura 21: Progressão da força $F_{ela,x}$ no tempo

Esta força, com o passar do tempo, passa então a adquirir valores cada vez mais negativos, até atingir o valor de $-1 \cdot 10^{-4}$ N. Observe que este valor é extremamente pequeno em módulo, muito distante dos valores obtidos com o uso do método da sobrerelaxação. Isto se deve, muito provavelmente ao fato de termos uma amostra limitada e diminuta do tempo. Este fenômeno já era observável nas análises anteriores do regime transiente fica mais clara aqui: para garantir estabilidade tomou-se um valor de Δt próximo de $2 \cdot 10^{-6}$ s. Assim, mesmo tendo um período de 500 intervalos de tempo, ele é insuficiente para a propagação do potencial das bobinas para a armadura. Sendo assim, é razoável se esperar que o valor da força eletromagnética neste ponto seja pequena e instável (pois é negativo).

6 Conclusões

A partir dos resultados coletados com os diferentes experimentos realizados, é possível identificar a força que o MDF (Método de Diferenças Finitas) tem para lidar com problemas em duas dimensões com malha bem definido. O estudo do problema dos eletroímãs para a levitação magnética de um trem demonstra ser um ótimo caso, e interdisciplinar, para avaliar a aplicação deste método para problemas de Engenharia real, em que existem malhas de formato não trivial, condições de contorno de difícil tratamento, e exigem análise crítica dos resultados obtidos, uma vez que o problema é de fato real, ou seja, todos os resultados obtidos neste exercício devem refletir os fenômenos observados no trem de levitação magnético físico.

A partir dos resultados observados com a variação da discretização, bem como os obtidos para o regime transiente, é possível perceber que o método adotado para implementar as formulações do MDF é extremamente sensível a discretização, ou seja, o sucesso da implementação obtido para a malha depende fortemente do valor de Δx adotado. Para valores pequenos, no entanto, percebe-se que o método de fato reproduz as características do fenômeno observado.

Outrossim, a oportunidade de produzir e testar as equações do MDF se mostrou um grande aprendizado aos alunos, uma vez que este desenvolvimento na prática exige grande base teórica sobre os métodos. Esta implementação forneceu aos alunos uma oportunidade de gerar resultados reais sem a necessidade de equipamentos para simulações físicas, se mostrando de grande importância para a fase de projeto de sistemas reais. Além disso, com este exercício foi possível entender e experienciar a importância do dimensionamento da discretização da malha em se tratando de problemas 2-D.

Apêndices

A Listing do script em Python

Listagem 1: Código fonte

```

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from mpl_toolkits.axes_grid1 import make_axes_locatable
import matplotlib
import os

# Parâmetros estéticos do matplotlib
plt.rcParams['mathtext.fontset'] = 'cm'
plt.rcParams['font.family'] = 'STIXGeneral'
params = {'legend.fontsize': 'x-large',
           'figure.figsize': (10, 5),
           'axes.labelsize': 'x-large',
           'axes.titlesize':'x-large',
           'xtick.labelsize':'x-large',
           'ytick.labelsize':'x-large'}
plt.rcParams.update(params)

class OOMFormatter(matplotlib.ticker.ScalarFormatter):
    def __init__(self, order=0, fformat="%1.1f", offset=True, mathText=True):
        self.oom = order
        self.fformat = fformat
        matplotlib.ticker.ScalarFormatter.__init__(
            self,useOffset=offset,useMathText=mathText
        )
    def _set_order_of_magnitude(self):
        self.orderOfMagnitude = self.oom
    def _set_format(self, vmin=None, vmax=None):
        self.format = self.fformat
        if self._useMathText:
            self.format = r'$\mathdefault{{}}$' % self.format

def create_folder(folder_name, path = os.getcwd()):
    """
    Função que cria uma pasta em um determinado diretório. Criada com o objetivo de
    ser usada junto com as funções de plotagem para salvar as figuras
    """

    Parameters
    -----
    folder_name: str
        nome da pasta a ser criada
    path: str (default=os.getcwd(), diretório de trabalho atual)
        diretório onde a pasta será criada
    """

    try:
        PATH = os.path.join(path, str(folder_name))
        os.mkdir(PATH)
        print("Folder", "{}".format(folder_name), "criado")
        return PATH
    except FileExistsError:

```

```

        return PATH
    except TypeError:
        print("TypeError between path {} and folder_name {}".format(
            path, folder_name
        ))

matplotlib.rcParams["savefig.directory"] = os.chdir(create_folder('Figures'))

def gera_grid(dx, l, h):
    """
    Função que cria, a partir das dimensões do espaço `l` e `h` e a partir da
    discretização deste espaço `dx` (= `dy`) um grid que o descreve

    Parameters
    -----
    dx: float
        passo
    l: float
        largura do grid
    h: float
        altura do grid

    Returns
    -----
    M: int
        tamanho da grade em x
    N: int
        tamanho da grade em y
    X: np.ndarray, dim=(N, M)
        meshgrid em x
    Y: np.ndarray, dim=(N, M)
        meshgrid em y
    grid: np.ndarray, dim=(N, M)
        grid do sistema, inicialmente em branco
    """
    # dimensões do grid
    M = int(l/dx) + 1
    N = int(h/dx) + 1

    # inicializando o grid
    grid = np.zeros((N, M))

    # criando dois vetores lineares x e y e os usando de base
    # para as matrizes X e Y, resultantes do meshgrid
    x = np.linspace(0, l, M)
    y = np.linspace(-h/2, h/2, N)
    X, Y = np.meshgrid(x, y)
    X = np.around(X, decimals=5)
    Y = np.around(Y, decimals=5)

    return M, N, X, Y, grid

def define_elementos(X_, Y_, grid):
    """

```

Função que toma o `grid`, gerado pelo método `gera_grid()` e o completa com a identificação dos elementos que o compõe cada qual em sua posição

Parameters

```
-----
X_ : np.ndarray, dim=(N, M)
    meshgrid em x
Y_ : np.ndarray, dim=(N, M)
    meshgrid em y
grid: np.ndarray, dim=(N, M)
    grid do sistema, inicialmente em branco
```

Notes

Para tornar o EP mais dinâmico e eficiente, decidiu-se criar apenas um grid composto por elementos infinitesimais que indicam o meio ao qual pertencem. Para tanto, utilizou-se a seguinte notação:

- 0: pontos que não fazem parte do grid (resto que fica acima e abaixo da parte à direita da bobina, para preencher o espaço retangular)
- 1: entreferro
- 2: armadura
- 3: núcleo de ferro
- 4: bobina_LE
- 5: bobina_LD

Com este método, todas as grandezas que dependem do meio podem facilmente ser definidas e, apenas com uma matriz, é possível descrever por exemplo a corrente Jz em todo o sistema (podemos filtrar os pontos em que temos a bobina e tornar todo o resto da matriz 0)

' ''

```
X = 100*X_
Y = 100*Y_
# definindo as regiões dos diferentes elementos
armadura = (X <= 4)
entreferro = np.logical_or(
    # primeiro retângulo,
    # -10 < y < 10
    # 4 < X < 5
    np.logical_and(X > 4, X < 5),
    # região do meio,
    # -6 < y < 6
    # 5 <= x < 14
    np.logical_and(
        np.logical_and(X >= 5, X < 14),
        np.logical_and(Y > -6, Y < 6)
    )
)

nucleo = np.logical_or(
    # pontas
    # y <= -6 / y >= 6
    # 5 <= x < 20
    np.logical_and(
        np.logical_and(X >= 5, X <= 20),
        np.logical_or(Y <= -6, Y >= 6)
    ),
    # barra vertical (entre as bobinas)
```

```

# -10 < y < 10
# 16 < x < 16
np.logical_and(X > 16, X < 20)
)

bobina_LE = np.logical_and(
    # bobina da esquerda (corrente entrando na folha)
    # -6 <= y <= 6
    # 14 <= x < 16
    np.logical_and(X >= 14, X <= 16),
    np.logical_and(Y > -6, Y < 6)
)

bobina_LD = np.logical_and(
    # bobina da direita (corrente saindo da folha)
    # -6 <= y <= 6
    # 20 <= x < 22
    np.logical_and(X >= 20, X <= 22),
    np.logical_and(Y > -6, Y < 6)
)

# assignando os códigos de cada região (regiões não assignadas não fazem
# parte do sistema, região externa, continuam em 0)
grid[entreferro] = 1
grid[armadura] = 2
grid[nucleo] = 3
grid[bobina_LE] = 4
grid[bobina_LD] = 5

return grid

def gera_mu(grid, urAr, urFex, urFey):
    '''
    Método que gera uma matriz com o valor das permeabilidades
    para cada elemento do grid

    Parameters
    -----
    grid: np.ndarray, dim=(N, M)
        grid do sistema, contém as identificações dos elementos
    urAr: float
        permeabilidade magnética do ar
    urFex: float
        permeabilidade magnética do ferro na direção x
    urFey: float
        permeabilidade magnética do ferro na direção y

    Returns
    -----
    mu: np.ndarray, dim=(N, M, 2)
        valores da permeabilidade (mu) em cada meio, com o primeiro eixo
        sendo x e o segundo sendo y
    '''

    mu = np.dstack((np.piecewise(grid,
        [

```

```

        grid==0,  #vazios      - mu_Ar
        grid==1,  #entreferro   - mu_Ar
        grid==2,  #armadura     - mu_Fex
        grid==3,  #nucleoFerro - mu_Fex
        grid==4,  #bobinaLE     - mu_Ar
        grid==5,  #bobinaLD     - mu_Ar
    ],
    [urAr, urAr, urFex, urFex, urAr, urAr]
),
np.piecewise(grid,
[
grid==0, #vazios      - mu_Ar
grid==1, #entreferro   - mu_Ar
grid==2, #armadura     - mu_Fey
grid==3, #nucleoFerro - mu_Fey
grid==4, #bobinaLE     - mu_Ar
grid==5, #bobinaLD     - mu_Ar
],
[urAr, urAr, urFey, urFey, urAr, urAr]
)))

```

return mu

def inicializa_matriz_A(dx, l, h):

'''

Função que inicializa a matriz A, dos potenciais eletromagnéticos

Parameters

dx: float
passo

l: float
largura do grid

h: float
altura do grid

'''

- , - , - , - , A = gera_grid(dx, l, h)

return A

def aplica_cc(A, grid):

'''

Função que aplica as condições de contorno em A, o que inclui as condições de Dirichlet (potencial na fronteira externa constante e igual a 0) e também zera o potencial para os pontos do grid que "não pertencem" ao problema (pontos que foram anexados para tornar) o grid regular, mas que não fazem parte do problema

'''

bordas com potencial nulo

A[0, :] = 0

A[-1, :] = 0

A[:, 0] = 0

A[:, -1] = 0

zera partes que não pertencem ao problema

A = np.where(grid==0, 0, A)

```

    return A

def gera_matriz_uJ(mu, X, Y, grid, transiente=False, dt=1, k=1, omega=60):
    """
    Função que gera a matriz de correntes, J, que é nula em todo
    o espaço, exceto dentro da bobina

    Parameters
    -----
    mu: np.ndarray, dim=(N, M, 2)
        valores da permeabilidade (mu) em cada meio
    X: np.ndarray, dim=(N, M)
        meshgrid em x
    Y: np.ndarray, dim=(N, M)
        meshgrid em y
    grid: np.ndarray, dim=(N, M)
        grid do sistema, inicialmente em branco
    transiente: bool (default=False)
        booleano de controle para saber se deve ser adicionado o termo
        transiente
    dt: int (default=1)
        discretização do tempo
    k: int (default=1)
        passo no tempo atual
    omega: float (default=60)
        valor da frequência da rede elétrica

    Returns
    -----
    uJ: np.ndarray, dim=(N, M)
    """

def calcula_J(y):
    """
    Função auxiliar que calcula a corrente dado um ponto y
    """
    return (2e6*np.cos(np.pi*y/12e-2) + 8e5)

J = np.where(grid==4, -calcula_J(Y), 0) + np.where(grid==5, calcula_J(Y), 0)
if transiente:
    uJ = J * np.cos(omega*k*dt)

else:
    uJ = np.multiply(J, mu[:, :, 0])

return uJ

def calcula_A(A, dx, M, N, grid, uJ, mu):
    """
    Função que calcula o potencial eletromagnético para
    todos os pontos do grid

    Parameters
    -----
    A: np.ndarray, dim=(N, M)
        matriz de potenciais eletromagnéticos da iteração anterior

```

```

dx: float
    discretização da malha
M: int
    dimensão da matriz A em x
N: int
    dimensão da matriz B em x
grid: np.ndarray, dim=(N, M)
    matriz que contém a identificação dos elementos
uJ: np.ndarray, dim=(N, M)
    matriz com os termos forçantes (correntes) multiplicados por mu
mu: np.ndarray, dim=(N, M, 2)
    valores da permeabilidade (mu) em cada meio

Returns
-----
Anew: np.ndarray, dim=(N, M)
    matriz de potenciais atualizada
'''

Anew = A.copy()
# para cada linha
for j in range(1, N-1):
    # para cada coluna
    for i in range(1, M-1):
        uC = mu[j + 1,      i, 1]
        uB = mu[j - 1,      i, 1]
        uE = mu[      j, i - 1, 0]
        uD = mu[      j, i + 1, 0]

        Anew[j, i] = 1/2 * (A[j+1, i]*(uB/(uC + uB)) + A[j-1, i]*(uC/(uC + uB))
                            + A[j, i+1]*(uE/(uD + uE)) + A[j, i-1]*(uD/(uD + uE))
                            + dx**2/2 * uJ[j, i])

aplica_cc(Anew, grid)
return Anew

def calcula_A_e1(A, dx, M, N, grid, uJ, mu):
    '''
    Função que calcula o potencial eletromagnético para
    todos os pontos do grid

Parameters
-----
A: np.ndarray, dim=(N, M)
    matriz de potenciais eletromagnéticos da iteração anterior
dx: float
    discretização da malha
M: int
    dimensão da matriz A em x
N: int
    dimensão da matriz B em x
grid: np.ndarray, dim=(N, M)
    matriz que contém a identificação dos elementos
uJ: np.ndarray, dim=(N, M)
    matriz com os termos forçantes (correntes) NÃO multiplicado
    pelos mus

```

```

mu: np.ndarray, dim=(N, M)
    valores da permeabilidade (mu) em cada meio

Returns
-----
Anew: np.ndarray, dim=(N, M)
    matriz de potenciais atualizada
'''

Anew = A.copy()
# para cada linha
for j in range(1, N-1):
    # para cada coluna
    for i in range(1, M-1):
        uC = mu[j + 1,      i,  1]
        uB = mu[j - 1,      i,  1]
        uE = mu[j,      i - 1, 0]
        uD = mu[j,      i + 1, 0]
        ux = mu[j,      i,  0]
        uy = mu[j,      i,  1]

        Anew[j, i] = 1/(ux + uy)*(
            ux*uE/(uE + uD)*A[j, i+1] + ux*uD/(uE + uD)*A[j, i-1]
            + uy*uB/(uB + uC)*A[j+1, i] + uy*uC/(uB + uC)*A[j-1, i]
            + ux*uy*dx**2*uJ[j, i]/2
        )
    aplica_cc(Anew, grid)
return Anew

def calcula_A_transiente(A, dx, M, N, grid, uJ, mu):
    '''
    Função que calcula o potencial eletromagnético para
    todos os pontos do grid

    Parameters
    -----
    A: np.ndarray, dim=(N, M)
        matriz de potenciais eletromagnéticos da iteração anterior
    dx: float
        discretização da malha
    M: int
        dimensão da matriz A em x
    N: int
        dimensão da matriz B em x
    grid: np.ndarray, dim=(N, M)
        matriz que contém a identificação dos elementos
    uJ: np.ndarray, dim=(N, M)
        matriz com os termos forçantes (correntes) NÃO multiplicado
        pelos mus
    mu: np.ndarray, dim=(N, M, 2)
        valores da permeabilidade (mu) em cada meio

    Returns
    -----
    Anew: np.ndarray, dim=(N, M)
        matriz de potenciais atualizada
    
```

```

    '''
Anew = A.copy()
# para cada linha
for j in range(1, N-1):
    # para cada coluna
    for i in range(1, M-1):
        uC = mu[j + 1,      i,  1]
        uB = mu[j - 1,      i,  1]
        uE = mu[j,      i - 1, 0]
        uD = mu[j,      i + 1, 0]
        ux = mu[j,      i,  0]
        uy = mu[j,      i,  1]

        # se estiver na bobina
        if grid[j, i] in [4, 5]:
            Anew[j, i] = 1/alfa * (
                ux*uE/(uE + uD)*A[j, i+1] + ux*uD/(uE + uD)*A[j, i-1]
                + uy*uB/(uB + uC)*A[j+1, i] + uy*uC/(uB + uC)*A[j-1, i]
                + (alfa - ux - uy)*A[j, i] + ux*uy*dx**2*uJ[j, i]/2
            )
        else:
            Anew[j, i] = 1/(ux + uy)*(
                ux*uE/(uE + uD)*A[j, i+1] + ux*uD/(uE + uD)*A[j, i-1]
                + uy*uB/(uB + uC)*A[j+1, i] + uy*uC/(uB + uC)*A[j-1, i]
            )
    aplica_cc(Anew, grid)
return Anew

def sobrerrelaxa(dx, l, h, iterative_func=calcula_A, max_iter=1e5):
    '''
    Função que aplica a sobre-relaxação usando com um determinado passo
    usando um determinado lambda até uma determinada tolerância

    Parameters
    -----
    dx: float
        passo
    l: float
        largura do grid
    h: float
        altura do grid
    iterative_func: callable (default=calcula_A)
        função iterativa pra
    max_iter: float (default=1e5)
        máximo de iterações que se deve fazer
    '''

    M, N, X, Y, grid = gera_grid(dx, l, h)
    A      = inicializa_matriz_A(dx, l, h)
    grid  = define_elementos(X, Y, grid)
    mu    = gera_mu(grid, urAr, urFex, urFey)

    uJ_parameters = {calcula_A: {'mu':mu, 'X':X, 'Y':Y, 'grid':grid},
                      calcula_A_el: {'mu':mu, 'X':X, 'Y':Y, 'grid':grid,
                                     'transiente':True, 'dt':1, 'k':1, 'omega':0}
                  }[iterative_func]

```

```

uJ      = gera_matriz_uJ(**uJ_parameters)
Anew = iterative_func(A, dx, M, N, grid, uJ, mu)
erro = np.max(np.abs(
    np.divide(Anew - A, Anew, out=np.zeros_like(Anew), where=Anew!=0)
))
i = 0
while erro > tol and i < max_iter:
    Anew, A = lbd*iterative_func(Anew, dx, M, N, grid, uJ, mu) + (1 - lbd)*A, Anew
    erro     = np.max(np.abs(np.divide(Anew - A, Anew, out=np.zeros_like(Anew), where=Anew!=0)))
    i+= 1
return Anew

def calcula_B(A, N, M, dx, mu):
    """
    Calcula o vetor B ponto a ponto e retorna um array que são as componentes em x e outro que são as componentes em y.

    Parameters
    -----
    A: np.ndarray, dim=(N, M)
        valores de Az
    N: int
        tamanho da grade em y
    M: int
        tamanho da grade em x
    dx: float
        discretização da malha
    mu: np.ndarray, dim=(N, M, 2)
        valores da permeabilidade (mu) em cada meio

    Returns
    -----
    Bx: np.ndarray, dim=(N, M)
        matriz das componentes de Bx
    By: np.ndarray, dim=(N, M)
        matriz das componentes de By
    """

    Bx = np.zeros_like(A)
    By = np.zeros_like(A)
    for j in range(1, N-1):
        for i in range(1, M-1):
            uC = mu[j + 1,      i,  1]
            uB = mu[j - 1,      i,  1]
            uE = mu[      j, i - 1, 0]
            uD = mu[      j, i + 1, 0]

            dela_delx = (A[j, i+1] - A[j, i-1])/(2*dx)
            dela_dely = (A[j+1, i] - A[j-1, i])/(2*dx)

            Bx[j, i] = dela_dely
            By[j, i] = - dela_delx

    return Bx, By

def calcula_H(B, mu):
    """

```

Função que calcula o vetor H , ou a componente relativa do vetor, a partir de um vetor B , ou de uma componente do vetor, e do vetor de mu

Parameters

B: np.ndarray, dim=(N, M)
matriz das componentes de B

mu: np.ndarray, dim=(N, M)
valores da permeabilidade (mu) em cada meio

Returns

H: np.ndarray, dim=(N, M)
valores da componente de H

Notes

Na hora de chamar colocar o mu[:, :, 0] para Bx e mu[:, :, 1] para By

"""

```
H = np.divide(B, mu)
return H
```

def calcula_F(Bx, By, grid, dx):
 """
 Função que calcula as forças nas direções x e y para a armadura.
 Ela é dada pela integral de linha seguindo o contorno na armadura, que
 neste caso foi discretizada em 4 partes.

Parameters

Bx: np.ndarray, dim=(N, M)
matriz das componentes de Bx

By: np.ndarray, dim=(N, M)
matriz das componentes de By

grid: np.ndarray, dim=(N, M)
grid do sistema, com os elementos definidos

dx: float
discretização

Returns

Fela_x: float
valor da componente x da força resultante do campo B

Fela_y: float
valor da componente y da força resultante do campo B

"""
max_y, max_x = max(np.where(grid == 2)[0]), max(np.where(grid == 2)[1])
min_y, min_x = min(np.where(grid == 2)[0]), min(np.where(grid == 2)[1])

Bx2_By2 = Bx**2 - By**2

BxBy = 2*np.multiply(Bx, By)

Fela_x = 1/(2*u0) * (
 np.trapz(Bx2_By2[min_y:max_y, max_x], dx=dx)
 - np.trapz(BxBy[max_y, min_x:max_x], dx=dx)
 - np.trapz(Bx2_By2[min_y:max_y, min_x], dx=dx))

```

        + np.trapz(      BxBy [      min_y ,  min_x:max_x] ,  dx=dx)
    )

Fela_y = 1/(2*u0) * (
    np.trapz(      BxBy[min_y:max_y ,           max_x] ,  dx=dx)
    + np.trapz(Bx2_By2[      max_y ,  min_x:max_x] ,  dx=dx)
    - np.trapz(      BxBy[min_y:max_y ,           min_x] ,  dx=dx)
    - np.trapz(Bx2_By2[      min_y ,  min_x:max_x] ,  dx=dx)
)

return Fela_x , Fela_y

def plot_surface(X, Y, A, exercise, instante=None):
    '''
    Função que realiza os plots de superfície (surface) e calor (heatmap)

    Parameters
    -----
    X: np.ndarray, dim=(N, M)
        meshgrid em x
    Y: np.ndarray, dim=(N, M)
        meshgrid em y
    A: np.ndarray, dim=(N, M)
        matriz dos potenciais eletromagnéticos
    instante: int (default=None)
        para plots de regime transiente, valor do instante k que se quer plotar
        se None, não especifica o tempo
    '''

    from mpl_toolkits.mplot3d import Axes3D
    from matplotlib import cm
    dx = X[0, 1] - X[0, 0]
    OM = '{:.1e}'.format(dx).split('e')
    if instante is None:
        ttl_str = r'$A_z$ para $\Delta x= %.1f \times 10^{%d}$ m
                    para o item %s)'%(float(OM[0]), int(OM[1]), exercise)
        sv_str = f'{exercise}_{dx}'
    else:
        ttl_str = r'$A_z$ para $\Delta x= %.1f \times 10^{%d}$ m
                    em $t=%d \Delta t$ para o item %s)'%
                    (float(OM[0]), int(OM[1]), instante, exercise)
        sv_str = f'{exercise}_{dx}_{instante}'

    plt.clf()
    fig, ax = plt.subplots(subplot_kw={"projection": "3d"})
    ax.view_init(elev=12., azim=-115)
    surf = ax.plot_surface(X, Y, A, cmap=cm.jet,
                           linewidth=0, antialiased=False)
    ax.xaxis.set_major_formatter(
        OOMFormatter(int('{:.2e}'.format(np.min(A)).split('e')[1]), mathText=True))
    ax.xaxis.set_rotate_label(False)
    ax.set_xlabel(r'$x$ (m)', size=20, labelpad=10)
    ax.set_ylabel(r'$y$ (m)', size=20, labelpad=10)
    ax.set_zlabel(r'$A_z$', size=20, labelpad=10)
    plt.tight_layout()

```

```

plt.title(ttl_str)
m = cm.ScalarMappable(cmap=cm.jet)
m.set_array(A)
plt.colorbar(m, aspect=8, pad=0,
    format=OOMFormatter(int(' {:.2e}'.format(np.min(A)).split('e')[1])), mathText=True)
)

plt.savefig(f'surface_{sv_str}.png', dpi=300, bbox_inches='tight')

plt.clf()
fig, ax = plt.subplots()

plt.imshow(A, origin='lower', interpolation="none", extent=[0, .22, -.10, .10], cmap=cm.jet)
ax = plt.gca()
ax.set_aspect(20/22)
ax.set_ylabel(r'$y$ (m)', size=20)
ax.set_xlabel(r'$x$ (m)', size=20)
cbar = plt.colorbar(format=OOMFormatter(int(' {:.2e}'.format(np.min(A)).split('e')[1])))
cbar.set_label(r'$A_z$ (T.m)', rotation=90, size=20)
plt.title(ttl_str)
plt.savefig(f'heatmap_{sv_str}.png', dpi=300, bbox_inches='tight')

def plot_vectorial_field(X, Y, Vx, Vy, name, exercise, instante=None):
    """
    Função que plota os campos vetoriais para a malha

    Parameters
    -----
    X: np.ndarray, dim=(N, M)
        meshgrid em x
    Y: np.ndarray, dim=(N, M)
        meshgrid em y
    Vx: np.ndarray, dim=(N, M)
        componentes do campo vetorial em x
    Vy: np.ndarray, dim=(N, M)
        componentes do campo vetorial em y
    name: str {'B', 'H'}
        o que se está plotando (B ou H)
    exercise: str {'c', 'e.1', 'e.2'}
        exercício ao qual se refere o plot, para colocar no
        título do gráfico e salvar com o nome correto
    instante: int (default=None)
        para plots de regime transiente, valor do instante k que se quer plotar
        se None, não especifica o tempo
    """

    plt.clf()
    unit = {'B': r'T', 'H': r'\frac{A}{m}'}[name]
    dx = X[0, 1] - X[0, 0]
    OM = ' {:.1e}'.format(dx).split('e')
    if instante is None:
        ttl_str = r'Campo vetorial de $%s$ para $\Delta x= %.1f \times 10^{%d}$ m
                    para o item %s)' %
                    (name, float(OM[0]), int(OM[1]), exercise
                    )
    else:
        ttl_str = r'Campo vetorial de $%s$ para $\Delta x= %.1f \times 10^{%d}$ m
                    para o item %s)' %
                    (name, float(OM[0]), int(OM[1]), exercise
                    )

```

```

        sv_str = f'{exercise}_{name}_{dx}'
    else:
        ttl_str = r'Campo vetorial de $%s$ para $\Delta x= %.1f \times 10^{%d}$ m
                    em $t=%d \Delta t$ para o item %s)'%(name, float(OM[0]), int(OM[1]), instante, exercise)
    )
    sv_str = f'{exercise}_{name}_{dx}_{instante}'

Q = plt.quiver(X, Y, Vx, Vy, color='red',
                width=0.0005, minshaft = 1, minlength=0.1
            )

plt.title(ttl_str)
ax = plt.gca()
ax.set_xlabel(r'$x$ (m)')
ax.set_ylabel(r'$y$ (m)')

plt.savefig('vfield_' + sv_str + '.png', dpi=300, bbox_inches='tight')

def plot_electromagnetic_force(Fela_vector, T, dx):
    '''
    Função que plota a evolução temporal da força eletromagnética no caso
    transiente

    Parameters
    -----
    Fela_vector: np.ndarray, dim=(T, 1)
        vetor que guarda os valores da força nos instantes requisitados
    T: int
        dimensão de Fela_vector
    dx: float
        discretização da malha
    '''

    plt.clf()
    OM = '{:.1e}'.format(dx).split('e')
    plt.plot([i for i in range(T)], Fela_vector)
    plt.grid()
    ax = plt.gca()
    ax.set_xlabel(r'$\Delta t$')
    ax.set_ylabel(r'$F_{ela, x};(N)$')
    plt.title(r'Força eletromagnética para $\Delta x= %.1f \times 10^{%d}$ $m$ para o item e.2)'%(float(OM[0]), int(OM[1])))

    plt.savefig('Felax_e.2.png', dpi=300, bbox_inches='tight')

# Parâmetros
u0      = 4*np.pi*1e-7 #... H/m
urFex   = 2500*u0 #..... H/m
urFey   = 2500*u0 #..... H/m
urAr    = 1*u0 #..... H/m
sBobina = 4e6 #... 1/0hm.m
l       = .22 #..... m
h       = .20 #..... m
lbd     = 1.75 # lambda, sobre relaxação
tol     = 1e-4 # tolerância, sobre relaxação

```

```

# itens A, B, C, D
dx_list = [5e-3, 2.5e-3, 1e-3]
for dx in dx_list:
    M, N, X, Y, grid = gera_grid(dx, l, h)
    grid = define_elementos(X, Y, grid)
    mu = gera_mu(grid, urAr, urFex, urFey)
    A = sobrerrelaxa(dx, l, h)
    plot_surface(X, Y, A, 'b')
    Bx, By = calcula_B(A, N, M, dx, mu)
    Hx = calcula_H(Bx, mu[:, :, 0])
    Hy = calcula_H(By, mu[:, :, 1])
    plot_vectorial_field(X, Y, Bx, By, 'B', 'c')
    plot_vectorial_field(X, Y, Hx, Hy, 'H', 'c')
    Fela_x, Fela_y = calcula_F(Bx, By, grid, dx)

# item E
# E.1
# Novos parâmetros
dx = 2.5e-1
urFex = 1200*u0

M, N, X, Y, grid = gera_grid(dx, l, h)
grid = define_elementos(X, Y, grid)
mu = gera_mu(grid, urAr, urFex, urFey)
A = sobrerrelaxa(dx, l, h, iterative_func=calcula_A_e1)
plot_surface(X, Y, A, 'e1')
Bx, By = calcula_B(A, N, M, dx, mu)
Hx = calcula_H(Bx, mu[:, :, 0])
Hy = calcula_H(By, mu[:, :, 1])
plot_vectorial_field(X, Y, Bx, By, 'B', 'e.1')
plot_vectorial_field(X, Y, Hx, Hy, 'H', 'e.1')
Fela_x, Fela_y = calcula_F(Bx, By, grid, dx)

# E.2
T = 501
M, N, X, Y, grid = gera_grid(dx, l, h)
A = inicializa_matriz_A(dx, l, h)
grid = define_elementos(X, Y, grid)
mu = gera_mu(grid, urAr, urFex, urFey)
dt = dx**2*urAr*sBobina/16
alfa = sBobina*urAr**2*dx**2/(2*dt)
Fela_x_vector = np.zeros(T)
for k in range(T):
    uJ = gera_matriz_uJ(mu, X, Y, grid, True, dt, k)
    A = calcula_A_transiente(A, dx, M, N, grid, uJ, mu)
    Bx, By = calcula_B(A, N, M, dx, mu)
    Hx = calcula_H(Bx, mu[:, :, 0])
    Hy = calcula_H(By, mu[:, :, 1])
    Fela_x, Fela_y = calcula_F(Bx, By, grid, dx)
    Fela_x_vector[k] = Fela_x
    if k in [10, 100, 500]:
        plot_surface(X, Y, A, 'e.2', k)
        plot_vectorial_field(X, Y, Bx, By, 'B', 'e.2', k)
        plot_vectorial_field(X, Y, Hx, Hy, 'H', 'e.2', k)
plot_electromagnetic_force(Fela_x_vector, T, dx)

```